CS418_ProgressReport_DeliveryAgainstDeserts

November 15, 2025

1 Food Delivery Against Food Deserts

1.1 1. Project Introduction

Food deserts are areas where residents have limited access to affordable and nutritious food. This problem affects approx. 39.5 million Americans (~11.6% of the US population). Online food and grocery delivery Stores seem to solve this problem to an extent, but issues of cost, digital access, and service availability mean not everyone benefits equally.

This project aims to **investigate the relationship between food accessibility, socioeconomic factors, and the potential of food delivery systems** in mitigating the effects of food deserts.

1.1.1 1.1 Why should others care?

- Public health impact
  - Food insecurity is linked to obesity, diabetes, cardiovascular disease, and other chronic conditions
  - COVID-19 accelerated digital food adoption, creating new opportunities and challenges
- Equity Considerations:
  - Only 37% of rural residents have access to major delivery services.
  - Cost, digital literacy, and broadband access create new forms of food inequality.
- Policy Implications
  - Supplemental Nutrition Assistance Program (SNAP) online purchasing expansion decisions
  - Mobile food market and delivery service integration strategies

1.1.2 Our Idea

- With the rise of online food delivery services, we want to investigate the effects of these digital services on food deserts, and whether these digital services can or have effectively bridge food access gaps.

1.1.3 Hypothesis

- H1: Food delivery coverage is negatively correlated with traditional food desert metrics
- H2: Delivery services improve food access but may not address affordability barriers for low income populations
- H3: Income decline in food desert regions, contrasted with inflation-adjusted growth in wealthier Chicago areas, has widened food access disparities and is likely to worsen them further.

- H4: Rural and elderly populations face greater barriers to digital food access despite service availability
- H5: The growth of online grocery delivery services may reduce reliance on local grocery stores, leading to store closures and creation of new physical food deserts for populations .

1.1.4 Data Sources:

- Food desert research access atlas
- SNAP benefits issuance data for Illinois
- FCC BroadBand Data
- Census Tract Data for Chicago
- Yelp Open dataset
- Grocery store coverage dataset by Chicago City Govt
- Amazon Delivery Dataset

1.2 2. Any Changes Since Proposal

haven't change much something something

[ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

[ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from sklearn.model_selection import train_test_split, cross_val_score,␣
 ↪StratifiedKFold
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report,␣
 ↪confusion_matrix, roc_auc_score, roc_curve, precision_score, recall_score,␣
 ↪f1_score
from sklearn.utils.class_weight import compute_sample_weight
import warnings
warnings.filterwarnings('ignore')

sns.set_style("whitegrid")
plt.rcParams['figure.figsize'] = (12, 7)

print("Setup complete!")
```

```
Setup complete!
```

1.3 3. Data Preparation

1.3.1 3.1. SNAP Bi-Annual State Project Area/County Level Participation and Issuance Data

[ ]:
```
SNAP_FILE = '/content/drive/MyDrive/CS418 Data Science Project/Combined folder for agent/Datasets/SNAP JAN 2025.xlsx'

# Load with correct header
snap_raw = pd.read_excel(SNAP_FILE, header=2)
new_columns = snap_raw.iloc[0].tolist()
snap_2025 = snap_raw[1:].copy()
snap_2025.columns = new_columns
snap_2025.reset_index(drop=True, inplace=True)

print(f" SNAP data loaded: {snap_2025.shape}")
```

```
SNAP data loaded: (2004, 14)
```

[ ]:
```
region_col = 'Substate/Region'

# Filter for Cook County
cook_mask = snap_2025[region_col].astype(str).str.contains('IL', na=False, case=False) & \
            snap_2025[region_col].astype(str).str.contains('COOK', na=False, case=False)
snap_cook = snap_2025[cook_mask]

print(f" Cook County SNAP records: {len(snap_cook)}")

# Helper function
def get_numeric_sum(df, col):
    if col in df.columns:
        vals = pd.to_numeric(df[col], errors='coerce')
        return vals.sum(skipna=True)
    else:
        print(f"  Column not found: {col}")
        return np.nan

# Extract metrics
snap_persons_col = 'Calc: SNAP Total PA and Non-PA People'
snap_households_col = 'Calc: SNAP Total PA and Non-PA Households'
snap_issuance_col = 'SNAP All Total Actual PA & Non-PA Issuance'

snap_2025_total = get_numeric_sum(snap_cook, snap_persons_col)
snap_2025_households = get_numeric_sum(snap_cook, snap_households_col)
```

```
snap_2025_issuance = get_numeric_sum(snap_cook, snap_issuance_col)

print(f"\nCook County SNAP (January 2025):")
print(f"   Total Participants: {snap_2025_total:,.0f}")
print(f"   Total Households: {snap_2025_households:,.0f}")
print(f"   Total Issuance: ${snap_2025_issuance:,.2f}")

if pd.notnull(snap_2025_total) and snap_2025_total > 0:
    avg_per_participant = snap_2025_issuance / snap_2025_total
    print(f"   Avg per Participant: ${avg_per_participant:.2f}")
```

```
 Cook County SNAP records: 27

Cook County SNAP (January 2025):
   Total Participants: 882,039
   Total Households: 507,246
   Total Issuance: $183,991,586.00
   Avg per Participant: $208.60
```

1.3.2 3.2. USDA - Food Access Research Atlas (2019) with TigerLine census Tract

[ ]:
```
FA_PATH = '/content/drive/MyDrive/CS418 Data Science Project/Dhwani_Notebook/
 ↪Food Access Research Atlas 2019.csv'

tracts = pd.read_csv(FA_PATH)
print(f"Food Atlas loaded: {tracts.shape}")

# Filter for Cook County
chicago = tracts[
    (tracts['State'] == 'Illinois') &
    (tracts['County'] == 'Cook County')
].copy()

print(f"Cook County tracts: {len(chicago)}")

# Data quality checks
print("\n Data Quality Report:")
print(f"   Tracts with missing SNAP data: {chicago['TractSNAP'].isna().sum()}")
print(f"   Tracts with zero population: {(chicago['Pop2010'] == 0).sum()}")
print(f"   Tracts with missing poverty data: {chicago['PovertyRate'].isna().
 ↪sum()}")

# Clean data
chicago = chicago[chicago['Pop2010'] > 0]
print(f"    Cleaned dataset: {len(chicago)} valid tracts")

# Download Illinois census tracts shapefile for mapping
```

```
!wget https://www2.census.gov/geo/tiger/TIGER2020/TRACT/tl_2020_17_tract.zip

# Unzip it
!unzip -o tl_2020_17_tract.zip

print("Shapefile downloaded and extracted!")
```

```
Food Atlas loaded: (72531, 147)
Cook County tracts: 1314

 Data Quality Report:
   Tracts with missing SNAP data: 0
   Tracts with zero population: 0
   Tracts with missing poverty data: 0
    Cleaned dataset: 1314 valid tracts
--2025-11-15 02:54:20--
https://www2.census.gov/geo/tiger/TIGER2020/TRACT/tl_2020_17_tract.zip
Resolving www2.census.gov (www2.census.gov)… 172.65.90.26, 172.65.90.25,
172.65.90.27, …
Connecting to www2.census.gov (www2.census.gov)|172.65.90.26|:443… connected.
HTTP request sent, awaiting response… 200 OK
Length: 9772970 (9.3M) [application/zip]
Saving to: ‘tl_2020_17_tract.zip.6’

tl_2020_17_tract.zi 100%[===================>]   9.32M  --.-KB/s    in 0.1s

2025-11-15 02:54:21 (73.7 MB/s) - ‘tl_2020_17_tract.zip.6’ saved
[9772970/9772970]

Archive:  tl_2020_17_tract.zip
 extracting: tl_2020_17_tract.cpg
  inflating: tl_2020_17_tract.dbf
  inflating: tl_2020_17_tract.prj
  inflating: tl_2020_17_tract.shp
  inflating: tl_2020_17_tract.shp.ea.iso.xml
  inflating: tl_2020_17_tract.shp.iso.xml
  inflating: tl_2020_17_tract.shx
Shapefile downloaded and extracted!
```

[ ]:
```
print("\n First 5 rows of Chicago data:")
print(chicago.head())

print("\n Basic Statistics:")
print(f"Low Income Tracts: {chicago['LowIncomeTracts'].sum()} out of␣
 ↪{len(chicago)}")
print(f"Low Access Tracts (1 mile): {chicago['LA1and10'].sum()} out of␣
 ↪{len(chicago)}")
```

```
print(f"Food Desert Tracts (Low Income + Low Access): {chicago[chicago['LILATracts_1And10'] == 1].shape[0]}")
```

```
 First 5 rows of Chicago data:
       CensusTract     State       County  Urban  Pop2010  OHU2010  \
20935  17031010100  Illinois  Cook County      1     4854     2302
20936  17031010201  Illinois  Cook County      1     6450     2463
20937  17031010202  Illinois  Cook County      1     2818     1115
20938  17031010300  Illinois  Cook County      1     6236     2826
20939  17031010400  Illinois  Cook County      1     5042     2098

       GroupQuartersFlag  NUMGQTRS  PCTGQTRS  LILATracts_1And10  …  \
20935                  0     218.0      4.49                  0  …
20936                  0     163.0      2.53                  0  …
20937                  0     315.0     11.18                  0  …
20938                  0     791.0     12.68                  0  …
20939                  0    1349.0     26.76                  0  …

       TractSeniors  TractWhite  TractBlack  TractAsian  TractNHOPI  \
20935         277.0      1810.0      2437.0       150.0         1.0
20936         314.0      2312.0      2350.0       299.0         6.0
20937         309.0      1237.0       953.0       143.0         1.0
20938         884.0      3267.0      1722.0       346.0        16.0
20939         263.0      3341.0       729.0       562.0         5.0

       TractAIAN  TractOMultir  TractHispanic  TractHUNV  TractSNAP
20935       33.0         423.0          616.0     1162.0      433.0
20936       48.0        1435.0         2049.0      751.0      851.0
20937       18.0         466.0          789.0      464.0      232.0
20938       21.0         864.0         1169.0      993.0      470.0
20939       17.0         388.0          480.0      655.0      277.0

[5 rows x 147 columns]

 Basic Statistics:
Low Income Tracts: 681 out of 1314
Low Access Tracts (1 mile): 168 out of 1314
Food Desert Tracts (Low Income + Low Access): 51
```

[ ]:
```
# Load the shapefile
import geopandas as gpd

# Read the shapefile
tracts_geo = gpd.read_file('tl_2020_17_tract.shp')

# Filter for Cook County only
```

```
cook_tracts = tracts_geo[tracts_geo['COUNTYFP'] == '031'].copy()

print(f" Cook County has {len(cook_tracts)} census tracts in the shapefile")

# Merge with our food desert data
# Make sure CensusTract format matches
chicago['GEOID'] = chicago['CensusTract'].astype(str)
cook_tracts['GEOID'] = cook_tracts['GEOID'].astype(str)

# Merge
chicago_geo = cook_tracts.merge(chicago, on='GEOID', how='inner')

print(f"Merged dataset has {len(chicago_geo)} tracts")

# Save the merged geographic dataset
chicago_geo.to_file('chicago_food_deserts_geo.geojson', driver='GeoJSON')

print(" All data prepared and ready for visualizations!")
```

```
 Cook County has 1332 census tracts in the shapefile
Merged dataset has 1284 tracts
 All data prepared and ready for visualizations!
```

[ ]:
```
# Create a summary statistics dataframe
summary_stats = pd.DataFrame({
    'Metric': [
        'Total Census Tracts',
        'Low Income Tracts',
        'Low Access Tracts (1 mile)',
        'Food Desert Tracts',
        'Urban Tracts',
        'Average Poverty Rate',
        'Median Family Income (avg)'
    ],
    'Value': [
        len(chicago),
        chicago['LowIncomeTracts'].sum(),
        chicago['LA1and10'].sum(),
        chicago['LILATracts_1And10'].sum(),
        chicago['Urban'].sum(),
        f"{chicago['PovertyRate'].mean():.1f}%",
        f"${chicago['MedianFamilyIncome'].mean():,.0f}"
    ]
})

print("\n" + "="*50)
print("COOK COUNTY (CHICAGO) FOOD DESERT SUMMARY - 2015")
```

```
print("="*50)
print(summary_stats.to_string(index=False))
print("="*50)

print("\n Ready to create visualizations!")
```

```
==================================================
COOK COUNTY (CHICAGO) FOOD DESERT SUMMARY - 2015
==================================================
                    Metric   Value
       Total Census Tracts    1314
         Low Income Tracts     681
Low Access Tracts (1 mile)     168
        Food Desert Tracts      51
              Urban Tracts    1312
      Average Poverty Rate   17.1%
Median Family Income (avg) $83,042
==================================================

 Ready to create visualizations!
```

[ ]:
```
snap_2019_total = chicago['TractSNAP'].sum()

print(f"SNAP Growth:")
print(f"   2019 Total: {snap_2019_total:,.0f}")
print(f"   2025 Total: {snap_2025_total:,.0f}")

snap_growth_rate = (snap_2025_total / snap_2019_total) - 1
snap_change = snap_2025_total - snap_2019_total

print(f"   Change: {snap_change:+,.0f} ({snap_growth_rate*100:+.2f}%)")

# Project to tracts
chicago['SNAP_2025_Projected'] = chicago['TractSNAP'] * (1 + snap_growth_rate)
chicago['SNAP_Change_2019_2025'] = chicago['SNAP_2025_Projected'] -␣
 ↪chicago['TractSNAP']
chicago['SNAP_Intensity_2019'] = (chicago['TractSNAP'] / chicago['Pop2010'] *␣
 ↪100).fillna(0)
chicago['SNAP_Intensity_2025'] = (chicago['SNAP_2025_Projected'] /␣
 ↪chicago['Pop2010'] * 100).fillna(0)
chicago['SNAP_Intensity_Change'] = chicago['SNAP_Intensity_2025'] -␣
 ↪chicago['SNAP_Intensity_2019']

print(f"\nSNAP 2025 projected for {len(chicago)} tracts")
```

```
SNAP Growth:
```

```
   2019 Total: 296,761
   2025 Total: 882,039
   Change: +585,278 (+197.22%)

SNAP 2025 projected for 1314 tracts
```

###3.3. Grocery Store Data

```
[ ]: # Load Grocery Store data
grocery_file = '/content/drive/MyDrive/CS418 Data Science Project/Combined folder for agent/Datasets/Grocery_Store_Status_-_Historical_20251114.csv'
```

1.4 4. Exploratory Data Analysis (EDA)

1.5 4.1. Visualizations and Hypotheses

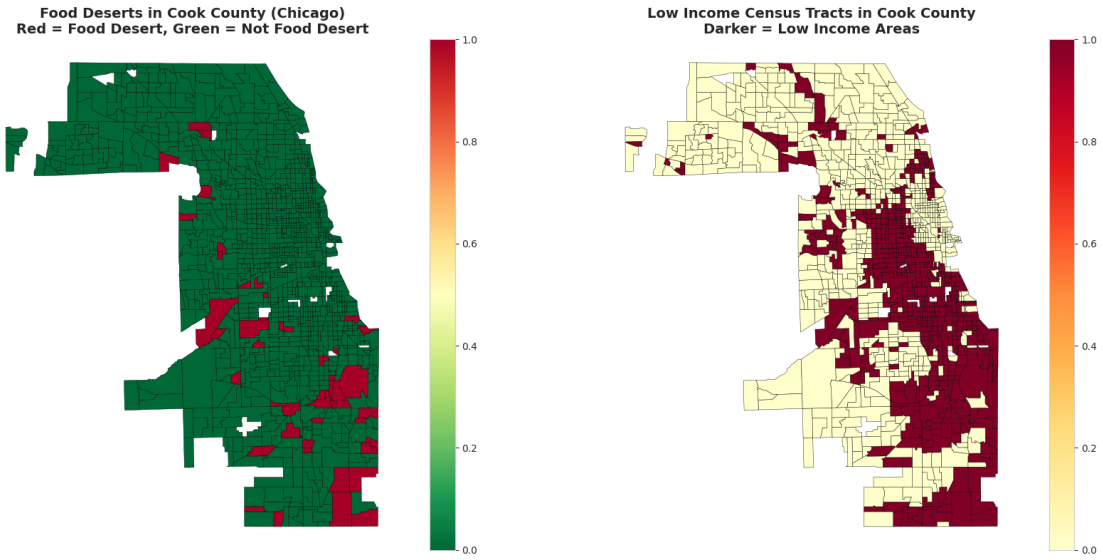
VISUALIZATION 1: Geographic Map of Food Deserts using Food desert dataset and Census Tract dataset - Dhwani Chande

Hypothesis: Food deserts cluster in specific geographic areas

```
[ ]: print("\n VISUALIZATION 1: Geographic Map of Food Deserts")
print("-" * 70)

# Set style for better-looking plots
sns.set_style("whitegrid")
plt.rcParams['figure.figsize'] = (12, 8)

fig, axes = plt.subplots(1, 2, figsize=(18, 8))

# Map 1: Food Desert Status
ax1 = axes[0]
chicago_geo.plot(column='LILATracts_1And10',
                 cmap='RdYlGn_r',
                 legend=True,
                 ax=ax1,
                 edgecolor='black',
                 linewidth=0.3)
ax1.set_title('Food Deserts in Cook County (Chicago)\nRed = Food Desert, Green = Not Food Desert',
              fontsize=14, fontweight='bold')
ax1.axis('off')

# Map 2: Low Income Tracts
ax2 = axes[1]
chicago_geo.plot(column='LowIncomeTracts',
                 cmap='YlOrRd',
                 legend=True,
                 ax=ax2,
```

```
                edgecolor='black',
                linewidth=0.3)
ax2.set_title('Low Income Census Tracts in Cook County\nDarker = Low Income␣
 ↪Areas',
              fontsize=14, fontweight='bold')
ax2.axis('off')

plt.tight_layout()
plt.show()

print("Hypothesis: Food deserts cluster in specific geographic areas")
print("Finding: Visual inspection shows concentration patterns in certain␣
 ↪regions")
print(f"   - {chicago['LILATracts_1And10'].sum()} food desert tracts out of␣
 ↪{len(chicago)} total")
print(f"   - {(chicago['LILATracts_1And10'].sum()/len(chicago)*100):.1f}% of␣
 ↪Cook County are food deserts")
```

```
 VISUALIZATION 1: Geographic Map of Food Deserts
--------------------------------------------------------------------------------
```



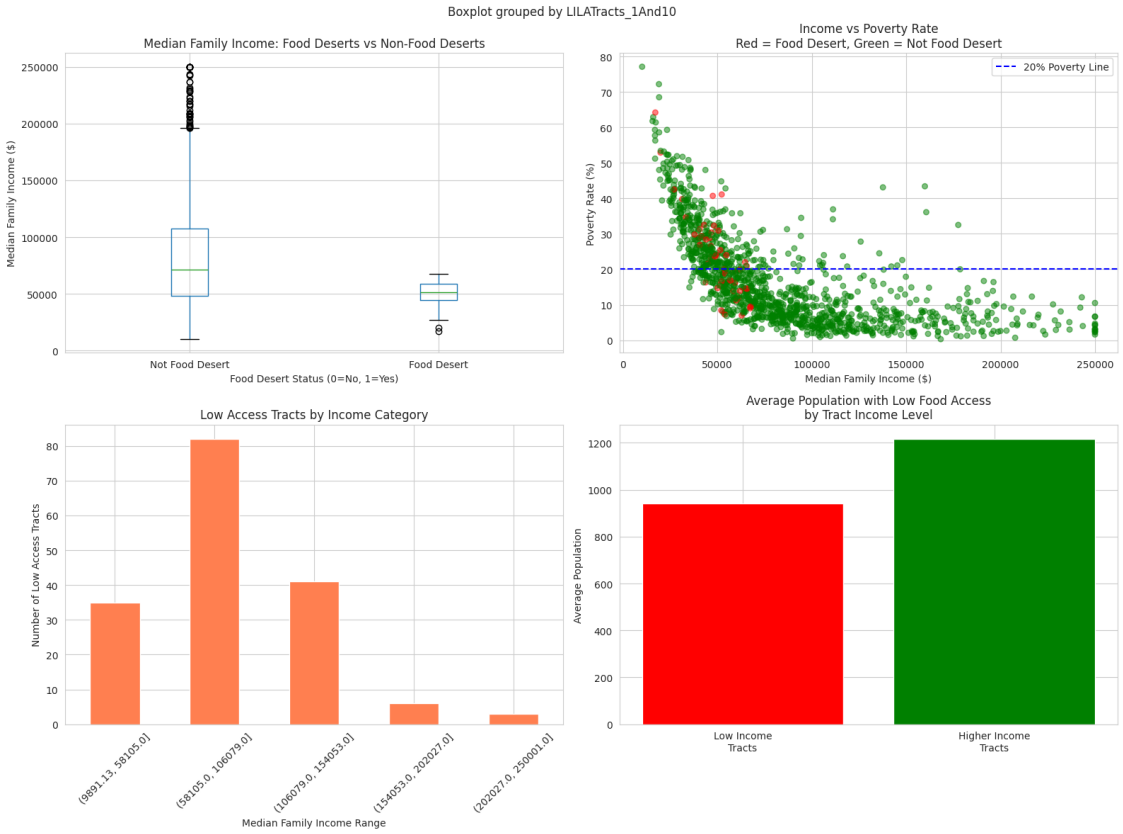

```
Hypothesis: Food deserts cluster in specific geographic areas
Finding: Visual inspection shows concentration patterns in certain regions
   - 51 food desert tracts out of 1314 total
   - 3.9% of Cook County are food deserts
```

VISUALIZATION 2: Income vs Food Access Analysis - Kaushik Sanjay Prabhakar
Hypothesis: Lower income correlates with reduced food access

[ ]:
```
print("\nVISUALIZATION 2: Income vs Food Access Analysis")
print("-" * 70)

fig, axes = plt.subplots(2, 2, figsize=(16, 12))

# Plot 1: Income distribution by food desert status
ax1 = axes[0, 0]
chicago.boxplot(column='MedianFamilyIncome', by='LILATracts_1And10', ax=ax1)
ax1.set_title('Median Family Income: Food Deserts vs Non-Food Deserts')
ax1.set_xlabel('Food Desert Status (0=No, 1=Yes)')
ax1.set_ylabel('Median Family Income ($)')
plt.sca(ax1)
plt.xticks([1, 2], ['Not Food Desert', 'Food Desert'])

# Plot 2: Scatter plot
ax2 = axes[0, 1]
colors = chicago['LILATracts_1And10'].map({0: 'green', 1: 'red'})
ax2.scatter(chicago['MedianFamilyIncome'], chicago['PovertyRate'],
           c=colors, alpha=0.5, s=30)
ax2.set_xlabel('Median Family Income ($)')
ax2.set_ylabel('Poverty Rate (%)')
ax2.set_title('Income vs Poverty Rate\nRed = Food Desert, Green = Not Food␣
 ↪Desert')
ax2.axhline(y=20, color='blue', linestyle='--', label='20% Poverty Line')
ax2.legend()

# Plot 3: Low access population by income category
ax3 = axes[1, 0]
income_bins = pd.cut(chicago['MedianFamilyIncome'], bins=5)
access_by_income = chicago.groupby(income_bins)['LA1and10'].sum()
access_by_income.plot(kind='bar', ax=ax3, color='coral')
ax3.set_title('Low Access Tracts by Income Category')
ax3.set_xlabel('Median Family Income Range')
ax3.set_ylabel('Number of Low Access Tracts')
ax3.tick_params(axis='x', rotation=45)

# Plot 4: Population in low access areas by income
ax4 = axes[1, 1]
low_income = chicago[chicago['LowIncomeTracts'] == 1]
high_income = chicago[chicago['LowIncomeTracts'] == 0]
data_compare = pd.DataFrame({
    'Category': ['Low Income\nTracts', 'Higher Income\nTracts'],
    'Avg Pop with Low Access': [
        low_income['LAPOP1_10'].mean(),
        high_income['LAPOP1_10'].mean()
    ]
})
```

```
ax4.bar(data_compare['Category'], data_compare['Avg Pop with Low Access'],
        color=['red', 'green'])
ax4.set_title('Average Population with Low Food Access\nby Tract Income Level')
ax4.set_ylabel('Average Population')

plt.tight_layout()
plt.show()

# Statistical analysis
food_desert_income = chicago[chicago['LILATracts_1And10'] ==␣
 ↪1]['MedianFamilyIncome'].mean()
non_food_desert_income = chicago[chicago['LILATracts_1And10'] ==␣
 ↪0]['MedianFamilyIncome'].mean()

print(" Hypothesis: Lower income correlates with reduced food access")
print(f" Finding: Food desert areas have significantly lower median income")
print(f"   - Food Desert areas: ${food_desert_income:,.0f} median income")
print(f"   - Non-Food Desert areas: ${non_food_desert_income:,.0f} median␣
 ↪income")
print(f"   - Income gap: ${(non_food_desert_income - food_desert_income):,.0f}")
```

```
VISUALIZATION 2: Income vs Food Access Analysis
--------------------------------------------------------------------------
```




```
Hypothesis: Lower income correlates with reduced food access
Finding: Food desert areas have significantly lower median income
  - Food Desert areas: $50,475 median income
  - Non-Food Desert areas: $84,339 median income
  - Income gap: $33,864
```

VISUALIZATION 3: Poverty Rate Distribution Analysis - Krishna Patel Hypothesis: Food deserts have higher poverty rates

[ ]:
```
print("\nVISUALIZATION 3: Poverty Rate Distribution Analysis")
print("-" * 70)

fig, axes = plt.subplots(2, 2, figsize=(16, 12))

# Plot 1: Poverty rate distribution
ax1 = axes[0, 0]
chicago[chicago['LILATracts_1And10'] == 0]['PovertyRate'].hist(
    bins=30, alpha=0.7, label='Non-Food Desert', color='green', ax=ax1)
chicago[chicago['LILATracts_1And10'] == 1]['PovertyRate'].hist(
    bins=30, alpha=0.7, label='Food Desert', color='red', ax=ax1)
ax1.set_xlabel('Poverty Rate (%)')
```

```
ax1.set_ylabel('Number of Census Tracts')
ax1.set_title('Poverty Rate Distribution: Food Deserts vs Non-Food Deserts')
ax1.legend()
ax1.axvline(x=20, color='blue', linestyle='--', label='20% Threshold')

# Plot 2: Poverty categories
ax2 = axes[0, 1]
poverty_categories = pd.cut(chicago['PovertyRate'],
                            bins=[0, 10, 20, 30, 100],
                            labels=['Low (<10%)', 'Medium (10-20%)',
                                    'High (20-30%)', 'Very High (>30%)'])
chicago['PovertyCategory'] = poverty_categories
poverty_food_desert = pd.crosstab(chicago['PovertyCategory'],
                                  chicago['LILATracts_1And10'],
                                  normalize='index') * 100
poverty_food_desert.plot(kind='bar', ax=ax2, color=['green', 'red'])
ax2.set_title('Percentage of Food Deserts by Poverty Category')
ax2.set_xlabel('Poverty Category')
ax2.set_ylabel('Percentage (%)')
ax2.legend(['Not Food Desert', 'Food Desert'])
ax2.tick_params(axis='x', rotation=45)

# Plot 3: Low income population in low access areas
ax3 = axes[1, 0]
poverty_levels = ['Low\n(<10%)', 'Medium\n(10-20%)', 'High\n(20-30%)', 'Very␣
 ↪High\n(>30%)']
avg_low_income_pop = []
for cat in poverty_categories.cat.categories:
    subset = chicago[chicago['PovertyCategory'] == cat]
    avg_low_income_pop.append(subset['LALOWI1_10'].mean())

ax3.bar(poverty_levels, avg_low_income_pop, color='coral')
ax3.set_title('Average Low-Income Population with Low Access\nby Poverty␣
 ↪Category')
ax3.set_xlabel('Poverty Category')
ax3.set_ylabel('Average Population')

# Plot 4: SNAP usage vs poverty
ax4 = axes[1, 1]
ax4.scatter(chicago['PovertyRate'], chicago['TractSNAP'],
           c=chicago['LILATracts_1And10'].map({0: 'green', 1: 'red'}),
           alpha=0.5, s=30)
ax4.set_xlabel('Poverty Rate (%)')
ax4.set_ylabel('SNAP Recipients (Count)')
ax4.set_title('SNAP Usage vs Poverty Rate\nRed = Food Desert, Green = Not Food␣
 ↪Desert')
```

```
plt.tight_layout()
plt.show()

food_desert_poverty = chicago[chicago['LILATracts_1And10'] == 1]['PovertyRate'].
  ↪mean()
non_food_desert_poverty = chicago[chicago['LILATracts_1And10'] ==␣
  ↪0]['PovertyRate'].mean()

print("Hypothesis: Food deserts have higher poverty rates")
print(f"Finding: Significant poverty rate differences observed")
print(f"   - Food Desert areas: {food_desert_poverty:.1f}% average poverty␣
  ↪rate")
print(f"   - Non-Food Desert areas: {non_food_desert_poverty:.1f}% average␣
  ↪poverty rate")
print(f"   - Poverty gap: {(food_desert_poverty - non_food_desert_poverty):.1f}␣
  ↪percentage points")
```

```
VISUALIZATION 3: Poverty Rate Distribution Analysis
--------------------------------------------------------------------------------
```

```
Hypothesis: Food deserts have higher poverty rates
Finding: Significant poverty rate differences observed
   - Food Desert areas: 24.0% average poverty rate
   - Non-Food Desert areas: 16.9% average poverty rate
   - Poverty gap: 7.1 percentage points
```

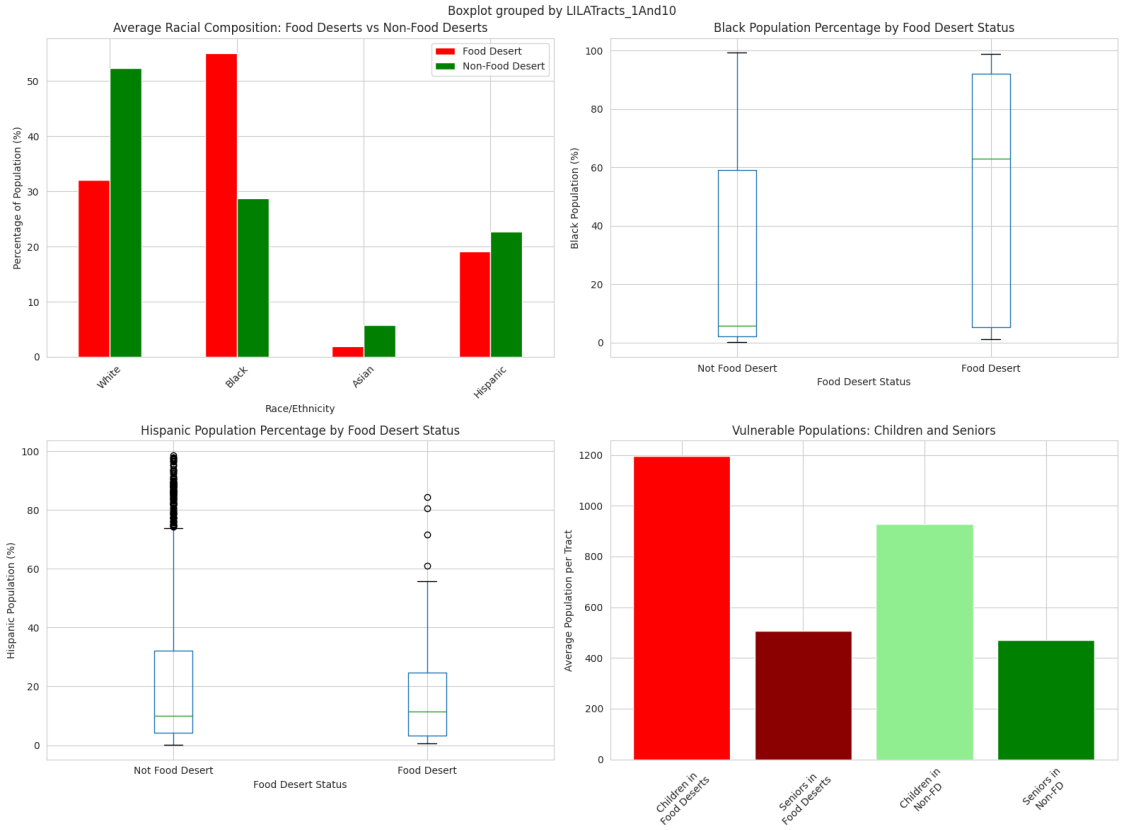
VISUALIZATION 4: Demographic Analysis of Food Deserts - Nishita Konduru **Hypothesis:** Food deserts disproportionately affect minority communities

[ ]:
```python
print("\nVISUALIZATION 4: Demographic Analysis of Food Deserts")
print("-" * 70)

fig, axes = plt.subplots(2, 2, figsize=(16, 12))

# Calculate racial percentages
chicago['PctWhite'] = (chicago['TractWhite'] / chicago['Pop2010']) * 100
chicago['PctBlack'] = (chicago['TractBlack'] / chicago['Pop2010']) * 100
chicago['PctAsian'] = (chicago['TractAsian'] / chicago['Pop2010']) * 100
chicago['PctHispanic'] = (chicago['TractHispanic'] / chicago['Pop2010']) * 100

# Plot 1: Demographic composition comparison
ax1 = axes[0, 0]
food_desert = chicago[chicago['LILATracts_1And10'] == 1]
non_food_desert = chicago[chicago['LILATracts_1And10'] == 0]

demographics_comparison = pd.DataFrame({
    'Food Desert': [
        food_desert['PctWhite'].mean(),
        food_desert['PctBlack'].mean(),
        food_desert['PctAsian'].mean(),
        food_desert['PctHispanic'].mean()
    ],
    'Non-Food Desert': [
        non_food_desert['PctWhite'].mean(),
        non_food_desert['PctBlack'].mean(),
        non_food_desert['PctAsian'].mean(),
        non_food_desert['PctHispanic'].mean()
    ]
}, index=['White', 'Black', 'Asian', 'Hispanic'])

demographics_comparison.plot(kind='bar', ax=ax1, color=['red', 'green'])
ax1.set_title('Average Racial Composition: Food Deserts vs Non-Food Deserts')
ax1.set_ylabel('Percentage of Population (%)')
ax1.set_xlabel('Race/Ethnicity')
ax1.legend()
ax1.tick_params(axis='x', rotation=45)
```

```
# Plot 2: Black population percentage in food deserts
ax2 = axes[0, 1]
chicago.boxplot(column='PctBlack', by='LILATracts_1And10', ax=ax2)
ax2.set_title('Black Population Percentage by Food Desert Status')
ax2.set_xlabel('Food Desert Status')
ax2.set_ylabel('Black Population (%)')
plt.sca(ax2)
plt.xticks([1, 2], ['Not Food Desert', 'Food Desert'])

# Plot 3: Hispanic population distribution
ax3 = axes[1, 0]
chicago.boxplot(column='PctHispanic', by='LILATracts_1And10', ax=ax3)
ax3.set_title('Hispanic Population Percentage by Food Desert Status')
ax3.set_xlabel('Food Desert Status')
ax3.set_ylabel('Hispanic Population (%)')
plt.sca(ax3)
plt.xticks([1, 2], ['Not Food Desert', 'Food Desert'])

# Plot 4: Senior population and children in food deserts
ax4 = axes[1, 1]
vulnerable_pops = pd.DataFrame({
    'Category': ['Children in\nFood Deserts', 'Seniors in\nFood Deserts',
                 'Children in\nNon-FD', 'Seniors in\nNon-FD'],
    'Average Population': [
        food_desert['TractKids'].mean(),
        food_desert['TractSeniors'].mean(),
        non_food_desert['TractKids'].mean(),
        non_food_desert['TractSeniors'].mean()
    ]
})
colors_vuln = ['red', 'darkred', 'lightgreen', 'green']
ax4.bar(vulnerable_pops['Category'], vulnerable_pops['Average Population'],
        color=colors_vuln)
ax4.set_title('Vulnerable Populations: Children and Seniors')
ax4.set_ylabel('Average Population per Tract')
ax4.tick_params(axis='x', rotation=45)

plt.tight_layout()
plt.show()

print("Hypothesis: Food deserts disproportionately affect minority communities")
print(f"Finding: Clear demographic disparities in food desert areas")
print(f"   - Black population in Food Deserts: {food_desert['PctBlack'].mean():.
 ↪1f}%")
print(f"   - Black population in Non-Food Deserts: {non_food_desert['PctBlack'].
 ↪mean():.1f}%")
```

```
print(f"   - Hispanic population in Food Deserts: {food_desert['PctHispanic'].
  ↪mean():.1f}%")
print(f"   - Hispanic population in Non-Food Deserts:␣
  ↪{non_food_desert['PctHispanic'].mean():.1f}%")
```

```
VISUALIZATION 4: Demographic Analysis of Food Deserts
----------------------------------------------------------------------
```




```
Hypothesis: Food deserts disproportionately affect minority communities
Finding: Clear demographic disparities in food desert areas
   - Black population in Food Deserts: 55.1%
   - Black population in Non-Food Deserts: 28.8%
   - Hispanic population in Food Deserts: 19.1%
   - Hispanic population in Non-Food Deserts: 22.8%
```

VISUALIZATION 5: Urban vs Distance to Supermarket Analysis - Anand Singh Kushwaha Hypothesis: Urban vs rural areas face different food access challenges"

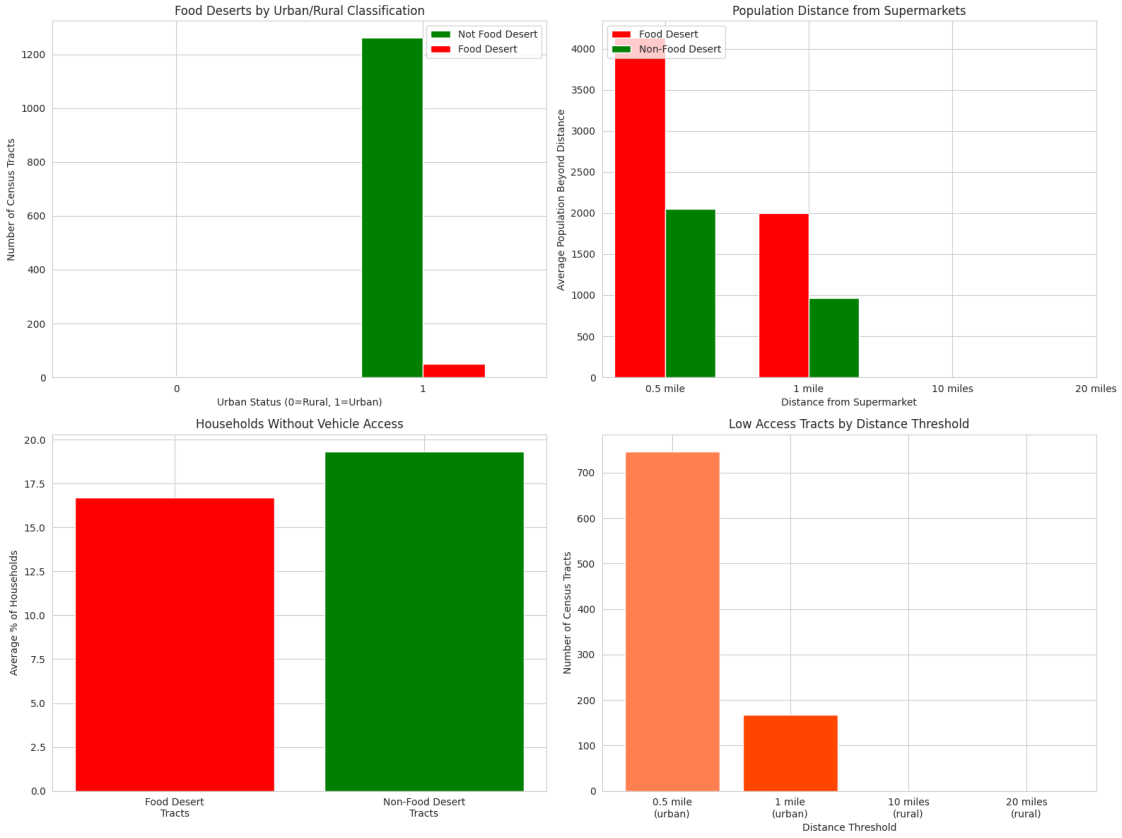
[ ]:
```
print("\nVISUALIZATION 5: Urban vs Distance to Supermarket Analysis")
print("-" * 70)
```

```
import numpy as np
import matplotlib.pyplot as plt

# Add the percentage column FIRST
chicago['PctNoVehicle'] = (chicago['TractHUNV'] / chicago['OHU2010']) * 100

# Now create the filtered subsets (so new column exists!)
food_desert = chicago[chicago['LILATracts_1And10'] == 1].copy()
non_food_desert = chicago[chicago['LILATracts_1And10'] == 0].copy()

# Confirm the column exists
print("PctNoVehicle in food_desert?", 'PctNoVehicle' in food_desert.columns)
print("PctNoVehicle in non_food_desert?", 'PctNoVehicle' in non_food_desert.
 ↪columns)

fig, axes = plt.subplots(2, 2, figsize=(16, 12))

# Plot 1: Urban vs Rural food desert distribution
ax1 = axes[0, 0]
urban_food_desert = pd.crosstab(chicago['Urban'], chicago['LILATracts_1And10'])
urban_food_desert.plot(kind='bar', ax=ax1, color=['green', 'red'])
ax1.set_title('Food Deserts by Urban/Rural Classification')
ax1.set_xlabel('Urban Status (0=Rural, 1=Urban)')
ax1.set_ylabel('Number of Census Tracts')
ax1.legend(['Not Food Desert', 'Food Desert'])
ax1.tick_params(axis='x', rotation=0)

# Plot 2: Population beyond different distances
ax2 = axes[0, 1]
distance_metrics = pd.DataFrame({
    'Distance': ['0.5 mile', '1 mile', '10 miles', '20 miles'],
    'Food Desert': [
        food_desert['LAPOP05_10'].mean(),
        food_desert['LAPOP1_10'].mean(),
        food_desert['lapop10'].mean(),
        food_desert['lapop20'].mean()
    ],
    'Non-Food Desert': [
        non_food_desert['LAPOP05_10'].mean(),
        non_food_desert['LAPOP1_10'].mean(),
        non_food_desert['lapop10'].mean(),
        non_food_desert['lapop20'].mean()
    ]
})
x = np.arange(len(distance_metrics['Distance']))
width = 0.35
```

```
ax2.bar(x - width/2, distance_metrics['Food Desert'], width,
        label='Food Desert', color='red')
ax2.bar(x + width/2, distance_metrics['Non-Food Desert'], width,
        label='Non-Food Desert', color='green')
ax2.set_xlabel('Distance from Supermarket')
ax2.set_ylabel('Average Population Beyond Distance')
ax2.set_title('Population Distance from Supermarkets')
ax2.set_xticks(x)
ax2.set_xticklabels(distance_metrics['Distance'])
ax2.legend()

# Plot 3: Vehicle access and food deserts
ax3 = axes[1, 0]
vehicle_access = pd.DataFrame({
    'Category': ['Food Desert\nTracts', 'Non-Food Desert\nTracts'],
    'Avg % No Vehicle': [
        food_desert['PctNoVehicle'].mean(),
        non_food_desert['PctNoVehicle'].mean()
    ]
})
ax3.bar(vehicle_access['Category'], vehicle_access['Avg % No Vehicle'],
        color=['red', 'green'])
ax3.set_title('Households Without Vehicle Access')
ax3.set_ylabel('Average % of Households')

# Plot 4: Low access at different thresholds
ax4 = axes[1, 1]
access_thresholds = pd.DataFrame({
    'Threshold': ['0.5 mile\n(urban)', '1 mile\n(urban)', '10 miles\n(rural)',␣
 ↪'20 miles\n(rural)'],
    'Number of Tracts': [
        chicago['LATracts_half'].sum(),
        chicago['LATracts1'].sum(),
        chicago['LATracts10'].sum(),
        chicago['LATracts20'].sum()
    ]
})
ax4.bar(access_thresholds['Threshold'], access_thresholds['Number of Tracts'],
        color=['coral', 'orangered', 'lightblue', 'steelblue'])
ax4.set_title('Low Access Tracts by Distance Threshold')
ax4.set_ylabel('Number of Census Tracts')
ax4.set_xlabel('Distance Threshold')

plt.tight_layout()
plt.show()

print("Hypothesis: Urban vs rural areas face different food access challenges")
```

```
print(f"Finding: Distance and vehicle access patterns differ significantly")
print(f"   - Urban tracts: {chicago['Urban'].sum()} out of {len(chicago)}")
print(f"   - Avg population beyond 1 mile (Food Desert): ␣
 ↪{food_desert['LAPOP1_10'].mean():.0f}")
print(f"   - Avg population beyond 1 mile (Non-FD): ␣
 ↪{non_food_desert['LAPOP1_10'].mean():.0f}")
print(f"   - % No vehicle (Food Desert): {food_desert['PctNoVehicle'].mean():.
 ↪1f}%")
print(f"   - % No vehicle (Non-FD): {non_food_desert['PctNoVehicle'].mean():.
 ↪1f}%")
```

```
VISUALIZATION 5: Urban vs Distance to Supermarket Analysis
--------------------------------------------------------------------------------
PctNoVehicle in food_desert? True
PctNoVehicle in non_food_desert? True
```




```
Hypothesis: Urban vs rural areas face different food access challenges
Finding: Distance and vehicle access patterns differ significantly
   - Urban tracts: 1312 out of 1314
   - Avg population beyond 1 mile (Food Desert): 2001
```

```
- Avg population beyond 1 mile (Non-FD): 970
- % No vehicle (Food Desert): 16.7%
- % No vehicle (Non-FD): 19.3%
```

1.6 5. Machine Learning Analysis

1.6.1 5.1. Machine Learning Analysis 1 - By Dhwani Chande (677725514)

In this section, we engineer demographic and economic features to predict food desert status using supervised machine learning. We'll train multiple classification models (Random Forest, Logistic Regression, Gradient Boosting, SVM, and Naive Bayes) to identify which socioeconomic factors most strongly influence food accessibility in Cook County.

1.6.2 Goals:

- Engineer 12 features from demographic, economic, and SNAP data
- Analyze feature correlations with food desert status
- Train and compare 5 classification models
- Evaluate model performance using accuracy, ROC-AUC, precision, recall, and F1-score
- Identify populations most affected by food deserts (children, seniors)
- Determine the top predictive features for food desert classification

[ ]:
```
print("\n" + "="*80)
print("FEATURE ENGINEERING")
print("="*80)

# Target
chicago['Food_Desert'] = chicago['LILATracts_1And10']

# Demographics
chicago['Pct_White'] = (chicago['TractWhite'] / chicago['Pop2010'] * 100).
 ↪fillna(0)
chicago['Pct_Black'] = (chicago['TractBlack'] / chicago['Pop2010'] * 100).
 ↪fillna(0)
chicago['Pct_Hispanic'] = (chicago['TractHispanic'] / chicago['Pop2010'] * 100).
 ↪fillna(0)
chicago['Pct_Asian'] = (chicago['TractAsian'] / chicago['Pop2010'] * 100).
 ↪fillna(0)
chicago['Pct_No_Vehicle'] = (chicago['TractHUNV'] / chicago['OHU2010'] * 100).
 ↪fillna(0)

# Economic
chicago['Income_Poverty_Ratio'] = (chicago['MedianFamilyIncome'] /␣
 ↪(chicago['PovertyRate'] * 100 + 1000))
chicago['Vulnerability_Index'] = ((chicago['TractKids'] +␣
 ↪chicago['TractSeniors']) / chicago['Pop2010'] * 100).fillna(0)

print("Feature engineering complete!")
```

```
# Define features
feature_cols = [
    'PovertyRate',
    'MedianFamilyIncome',
    'Income_Poverty_Ratio',
    'Pct_White',
    'Pct_Black',
    'Pct_Hispanic',
    'Pct_Asian',
    'Pct_No_Vehicle',
    'SNAP_Intensity_2019',
    'SNAP_Intensity_2025',
    'SNAP_Intensity_Change',
    'Vulnerability_Index'
]

ml_data = chicago[feature_cols + ['Food_Desert']].copy()
ml_clean = ml_data.dropna()

print(f"ML Dataset: {len(ml_clean)} samples × {len(feature_cols)} features")
print(f"\nClass Distribution:")
print(ml_clean['Food_Desert'].value_counts())
print(ml_clean['Food_Desert'].value_counts(normalize=True))
print("\n" + "="*80)
print("FEATURE CORRELATION ANALYSIS")
print("="*80)

correlations = ml_clean[feature_cols].corrwith(ml_clean['Food_Desert']).
 ↪sort_values(ascending=False)
print("\nCorrelation with Food Desert Status:")
print(correlations.to_string())

plt.figure(figsize=(10, 6))
colors = ['green' if x > 0 else 'red' for x in correlations]
correlations.plot(kind='barh', color=colors, edgecolor='black')
plt.xlabel('Correlation with Food Desert', fontweight='bold', fontsize=12)
plt.title('Feature Correlations with Food Desert Status', fontweight='bold',␣
 ↪fontsize=14)
plt.grid(alpha=0.3, axis='x')
plt.tight_layout()
plt.show()

print("\n" + "="*80)
print("TRAIN-TEST SPLIT")
print("="*80)
```

```
X = ml_clean[feature_cols]
y = ml_clean['Food_Desert']

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42, stratify=y
)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

print(f"Train set: {len(X_train)} samples")
print(f"Test set: {len(X_test)} samples")

print("\n" + "="*80)
print("TRAIN RANDOM FOREST MODEL")
print("="*80)

rf_model = RandomForestClassifier(
    n_estimators=200,
    max_depth=10,
    min_samples_split=10,
    class_weight='balanced',
    random_state=42
)

rf_model.fit(X_train, y_train)

y_pred_rf = rf_model.predict(X_test)
y_proba_rf = rf_model.predict_proba(X_test)[:, 1]

accuracy_rf = accuracy_score(y_test, y_pred_rf)
roc_auc_rf = roc_auc_score(y_test, y_proba_rf)

print(f"Random Forest Results:")
print(f"   Accuracy: {accuracy_rf:.3f}")
print(f"   ROC-AUC: {roc_auc_rf:.3f}")
print(f"\n{classification_report(y_test, y_pred_rf, target_names=['Not FD',␣
 ↪'Food Desert'])}")

cm = confusion_matrix(y_test, y_pred_rf)

plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Not FD', 'Food Desert'],
            yticklabels=['Not FD', 'Food Desert'])
plt.ylabel('Actual', fontweight='bold')
```

```
plt.xlabel('Predicted', fontweight='bold')
plt.title('Confusion Matrix - Random Forest', fontsize=14, fontweight='bold')
plt.tight_layout()
plt.show()

feature_importance = pd.DataFrame({
    'Feature': feature_cols,
    'Importance': rf_model.feature_importances_
}).sort_values('Importance', ascending=False)

plt.figure(figsize=(10, 6))
plt.barh(range(len(feature_importance)), feature_importance['Importance'],
         color='skyblue', edgecolor='black')
plt.yticks(range(len(feature_importance)), feature_importance['Feature'])
plt.xlabel('Importance', fontweight='bold', fontsize=12)
plt.title('Random Forest Feature Importances', fontweight='bold', fontsize=14)
plt.gca().invert_yaxis()
plt.grid(alpha=0.3, axis='x')
plt.tight_layout()
plt.show()

print("\nTop 5 Features:")
print(feature_importance.head().to_string(index=False))

print("\n" + "="*80)
print("MULTI-MODEL COMPARISON")
print("="*80)

# Define models
models = {
    'Random Forest': RandomForestClassifier(
        n_estimators=200,
        max_depth=10,
        min_samples_split=10,
        class_weight='balanced',
        random_state=42
    ),
    'Logistic Regression': LogisticRegression(
        random_state=42,
        max_iter=2000,
        class_weight='balanced',
        solver='liblinear'
    ),
    'Gradient Boosting': GradientBoostingClassifier(
        n_estimators=100,
        learning_rate=0.05,
        max_depth=5,
```

```
        min_samples_split=10,
        random_state=42
    ),
    'SVM (RBF)': SVC(
        kernel='rbf',
        probability=True,
        class_weight='balanced',
        random_state=42
    ),
    'Naive Bayes': GaussianNB()
}

# Calculate sample weights for Gradient Boosting
sample_weights_train = compute_sample_weight(class_weight='balanced', y=y_train)

results = []

print("\nTraining models...")
for name, model in models.items():
    print(f"   Training {name}...")

    if name == 'Gradient Boosting':
        model.fit(X_train, y_train, sample_weight=sample_weights_train)
        y_pred = model.predict(X_test)
        y_proba = model.predict_proba(X_test)[:, 1]
    elif name in ['Logistic Regression', 'SVM (RBF)']:
        model.fit(X_train_scaled, y_train)
        y_pred = model.predict(X_test_scaled)
        y_proba = model.predict_proba(X_test_scaled)[:, 1]
    else:
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)
        y_proba = model.predict_proba(X_test)[:, 1]

    acc = accuracy_score(y_test, y_pred)
    roc = roc_auc_score(y_test, y_proba)
    precision = precision_score(y_test, y_pred, zero_division=0)
    recall = recall_score(y_test, y_pred, zero_division=0)
    f1 = f1_score(y_test, y_pred, zero_division=0)

    cm = confusion_matrix(y_test, y_pred)
    tn, fp, fn, tp = cm.ravel()

    results.append({
        'Model': name,
        'Accuracy': acc,
        'ROC-AUC': roc,
```

```
        'Precision': precision,
        'Recall': recall,
        'F1-Score': f1,
        'True_Pos': tp,
        'False_Pos': fp,
        'False_Neg': fn,
        'True_Neg': tn
    })

results_df = pd.DataFrame(results).sort_values('ROC-AUC', ascending=False)

print("\nModel Performance:")
print(results_df[['Model', 'Accuracy', 'ROC-AUC', 'Precision', 'Recall',␣
 ↪'F1-Score']].to_string(index=False))

fig, axes = plt.subplots(2, 2, figsize=(16, 12))

# Plot 1: ROC-AUC
ax1 = axes[0, 0]
x_pos = np.arange(len(results_df))
width = 0.35

ax1.bar(x_pos - width/2, results_df['Accuracy'], width, label='Accuracy',␣
 ↪color='skyblue', edgecolor='black')
ax1.bar(x_pos + width/2, results_df['ROC-AUC'], width, label='ROC-AUC',␣
 ↪color='coral', edgecolor='black')
ax1.set_xticks(x_pos)
ax1.set_xticklabels(results_df['Model'], rotation=45, ha='right')
ax1.set_ylabel('Score', fontweight='bold')
ax1.set_title('Accuracy vs ROC-AUC', fontweight='bold')
ax1.legend()
ax1.grid(alpha=0.3, axis='y')
ax1.set_ylim(0.5, 1.0)

# Plot 2: Precision-Recall-F1
ax2 = axes[0, 1]
width2 = 0.25

ax2.bar(x_pos - width2, results_df['Precision'], width2, label='Precision',␣
 ↪color='green', edgecolor='black')
ax2.bar(x_pos, results_df['Recall'], width2, label='Recall', color='orange',␣
 ↪edgecolor='black')
ax2.bar(x_pos + width2, results_df['F1-Score'], width2, label='F1-Score',␣
 ↪color='purple', edgecolor='black')
ax2.set_xticks(x_pos)
ax2.set_xticklabels(results_df['Model'], rotation=45, ha='right')
```

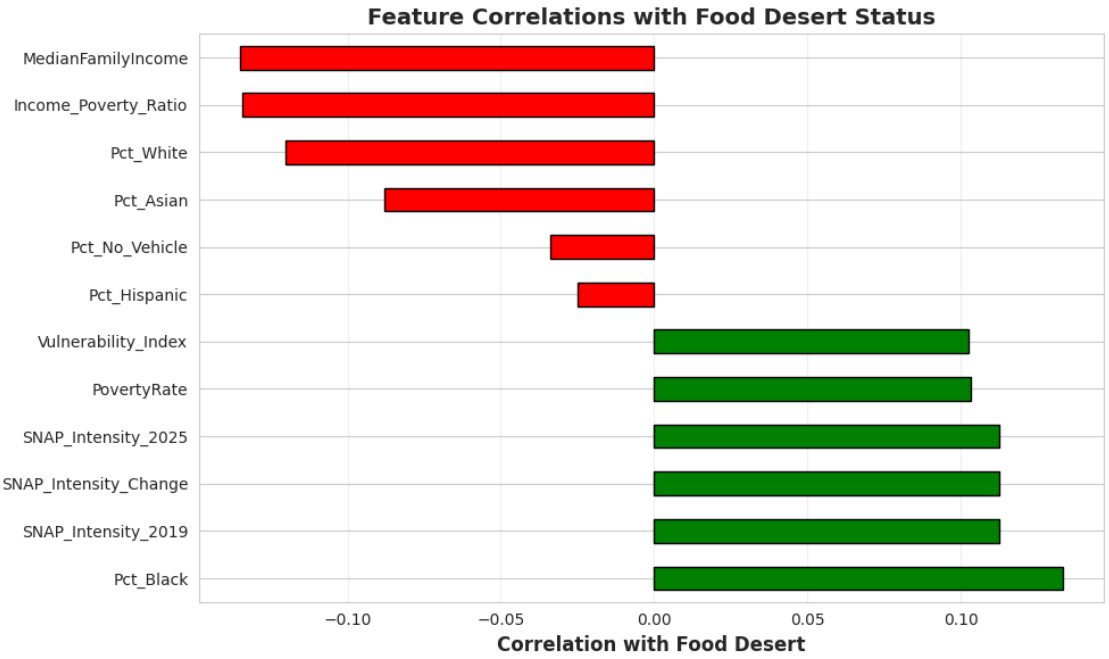
```
ax2.set_ylabel('Score', fontweight='bold')
ax2.set_title('Precision, Recall, F1-Score', fontweight='bold')
ax2.legend()
ax2.grid(alpha=0.3, axis='y')

# Plot 3: TP vs FN
ax3 = axes[1, 0]
ax3.bar(x_pos - width/2, results_df['True_Pos'], width, label='Caught',␣
 ↪color='#2ecc71', edgecolor='black')
ax3.bar(x_pos + width/2, results_df['False_Neg'], width, label='Missed',␣
 ↪color='#e74c3c', edgecolor='black')
ax3.set_xticks(x_pos)
ax3.set_xticklabels(results_df['Model'], rotation=45, ha='right')
ax3.set_ylabel('Count', fontweight='bold')
ax3.set_title('Food Deserts: Caught vs Missed', fontweight='bold')
ax3.legend()
ax3.grid(alpha=0.3, axis='y')

# Plot 4: ROC Curves
ax4 = axes[1, 1]
for name, model in models.items():
    if name == 'Gradient Boosting':
        y_proba = model.predict_proba(X_test)[:, 1]
    elif name in ['Logistic Regression', 'SVM (RBF)']:
        y_proba = model.predict_proba(X_test_scaled)[:, 1]
    else:
        y_proba = model.predict_proba(X_test)[:, 1]

    fpr, tpr, _ = roc_curve(y_test, y_proba)
    auc = roc_auc_score(y_test, y_proba)
    ax4.plot(fpr, tpr, label=f'{name} ({auc:.3f})', linewidth=2)

ax4.plot([0, 1], [0, 1], 'k--', linewidth=1)
ax4.set_xlabel('False Positive Rate', fontweight='bold')
ax4.set_ylabel('True Positive Rate', fontweight='bold')
ax4.set_title('ROC Curves', fontweight='bold')
ax4.legend(loc='lower right', fontsize=9)
ax4.grid(alpha=0.3)

plt.tight_layout()
plt.show()

best_model = results_df.iloc[0]
best_recall = results_df.loc[results_df['Recall'].idxmax()]

print(f"\nBEST MODEL (ROC-AUC): {best_model['Model']}")
print(f"   ROC-AUC: {best_model['ROC-AUC']:.3f}")
```

```
print(f"   Recall: {best_model['Recall']:.3f}")

print(f"\nBEST RECALL: {best_recall['Model']}")
print(f"   Recall: {best_recall['Recall']:.3f} (catches␣
 ↪{best_recall['Recall']*100:.0f}% of food deserts)")

print("\n" + "="*80)
print("POPULATION AFFECTED ANALYSIS")
print("="*80)

pop_affected = pd.DataFrame({
    'Food Desert': [
        chicago[chicago['Food_Desert']==1]['TractKids'].sum(),
        chicago[chicago['Food_Desert']==1]['TractSeniors'].sum(),
        chicago[chicago['Food_Desert']==1]['Pop2010'].sum()
    ],
    'Has Access': [
        chicago[chicago['Food_Desert']==0]['TractKids'].sum(),
        chicago[chicago['Food_Desert']==0]['TractSeniors'].sum(),
        chicago[chicago['Food_Desert']==0]['Pop2010'].sum()
    ]
}, index=['Children', 'Seniors', 'Total Pop'])

print(pop_affected)

fig, ax = plt.subplots(figsize=(12, 6))
pop_affected.plot(kind='barh', ax=ax, color=['#d62728', '#2ca02c'],␣
 ↪edgecolor='black', linewidth=1.5)
ax.set_xlabel('Population', fontweight='bold', fontsize=12)
ax.set_title('Population Distribution: Food Desert vs Access',␣
 ↪fontweight='bold', fontsize=14)
ax.legend(['Food Desert', 'Has Access'])
ax.grid(alpha=0.3, axis='x')
plt.tight_layout()
plt.show()

print("\n" + "="*80)
print("FINAL ANALYSIS SUMMARY")
print("="*80)

print(f"\nDataset:")
print(f"   Cook County tracts: {len(chicago)}")
print(f"   Food deserts: {(ml_clean['Food_Desert']==1).sum()}␣
 ↪({(ml_clean['Food_Desert']==1).sum()/len(ml_clean)*100:.1f}%)")
print(f"   Population in food deserts:␣
 ↪{chicago[chicago['Food_Desert']==1]['Pop2010'].sum():,.0f}")
```

```
print(f"\nSNAP Trends:")
print(f"   2019: {snap_2019_total:,.0f}")
print(f"   2025: {snap_2025_total:,.0f}")
print(f"   Growth: {snap_growth_rate*100:+.1f}%")

print(f"\nBest Models:")
print(f"   ROC-AUC: {results_df.iloc[0]['Model']} ({results_df.
 ↪iloc[0]['ROC-AUC']:.3f})")
print(f"   Recall: {results_df.loc[results_df['Recall'].idxmax()]['Model']}␣
 ↪({results_df['Recall'].max():.3f})")

print(f"\nTop 3 Features:")
for i, row in feature_importance.head(3).iterrows():
    print(f"   {i+1}. {row['Feature']}: {row['Importance']:.3f}")

print("\nANALYSIS COMPLETE!")
```

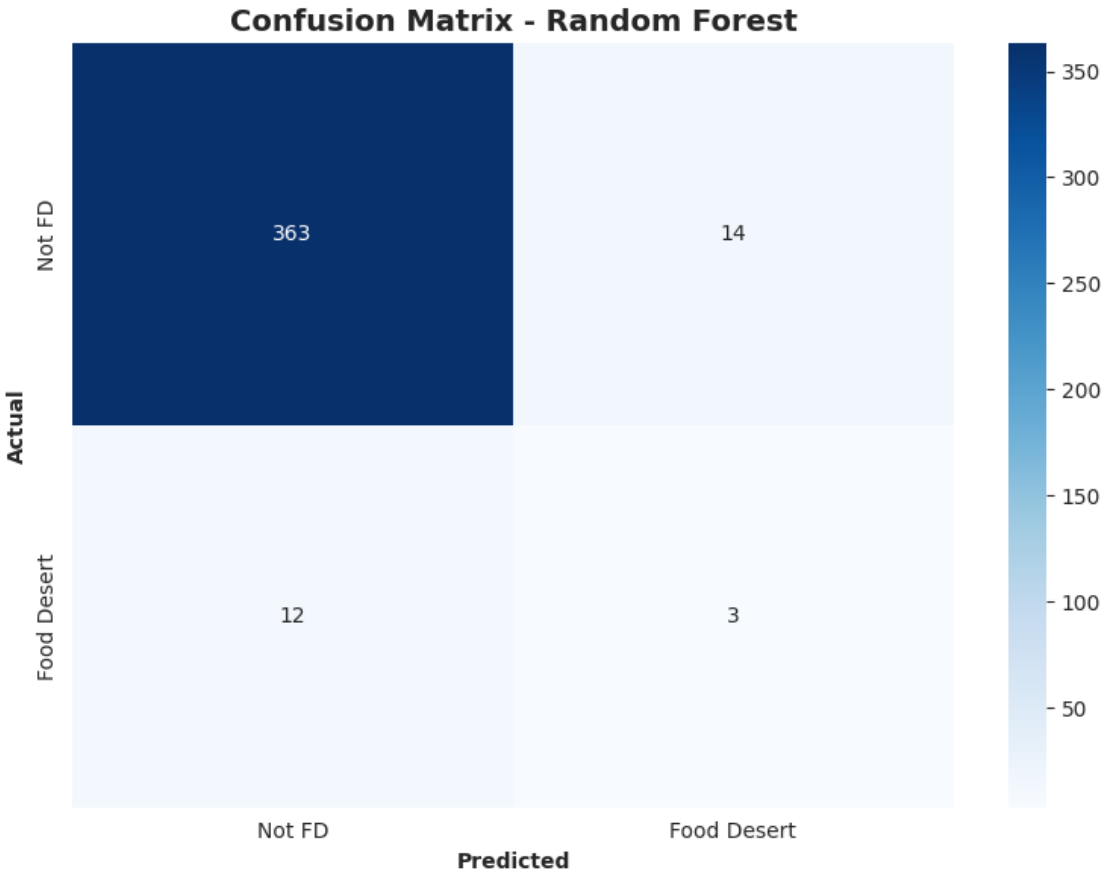
```
================================================================================
FEATURE ENGINEERING
================================================================================
Feature engineering complete!
ML Dataset: 1305 samples × 12 features

Class Distribution:
Food_Desert
0    1255
1      50
Name: count, dtype: int64
Food_Desert
0    0.961686
1    0.038314
Name: proportion, dtype: float64

================================================================================
FEATURE CORRELATION ANALYSIS
================================================================================

Correlation with Food Desert Status:
Pct_Black                0.133284
SNAP_Intensity_2019      0.112522
SNAP_Intensity_Change    0.112522
SNAP_Intensity_2025      0.112522
PovertyRate              0.103443
Vulnerability_Index      0.102396
Pct_Hispanic            -0.024971
Pct_No_Vehicle          -0.033675
```

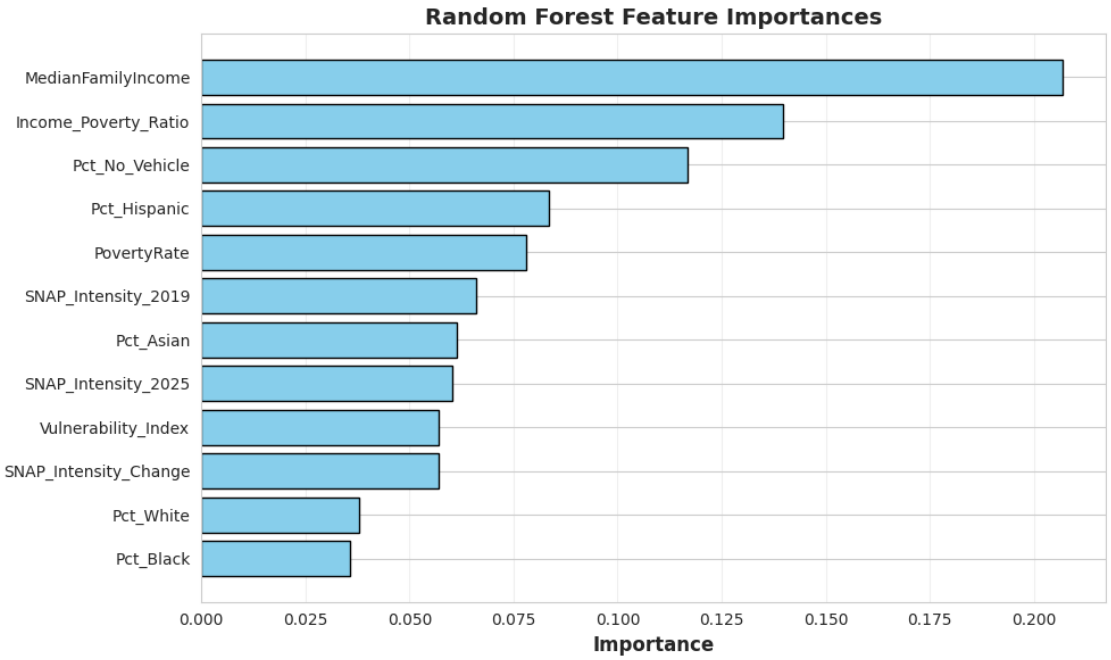
```
Pct_Asian               -0.088145
Pct_White               -0.120395
Income_Poverty_Ratio    -0.134254
MedianFamilyIncome      -0.134908
```




```
================================================================================
TRAIN-TEST SPLIT
================================================================================
Train set: 913 samples
Test set: 392 samples

================================================================================
TRAIN RANDOM FOREST MODEL
================================================================================
Random Forest Results:
   Accuracy: 0.934
   ROC-AUC: 0.803

              precision    recall  f1-score   support

      Not FD       0.97      0.96      0.97       377
 Food Desert       0.18      0.20      0.19        15

    accuracy                           0.93       392
   macro avg       0.57      0.58      0.58       392
```

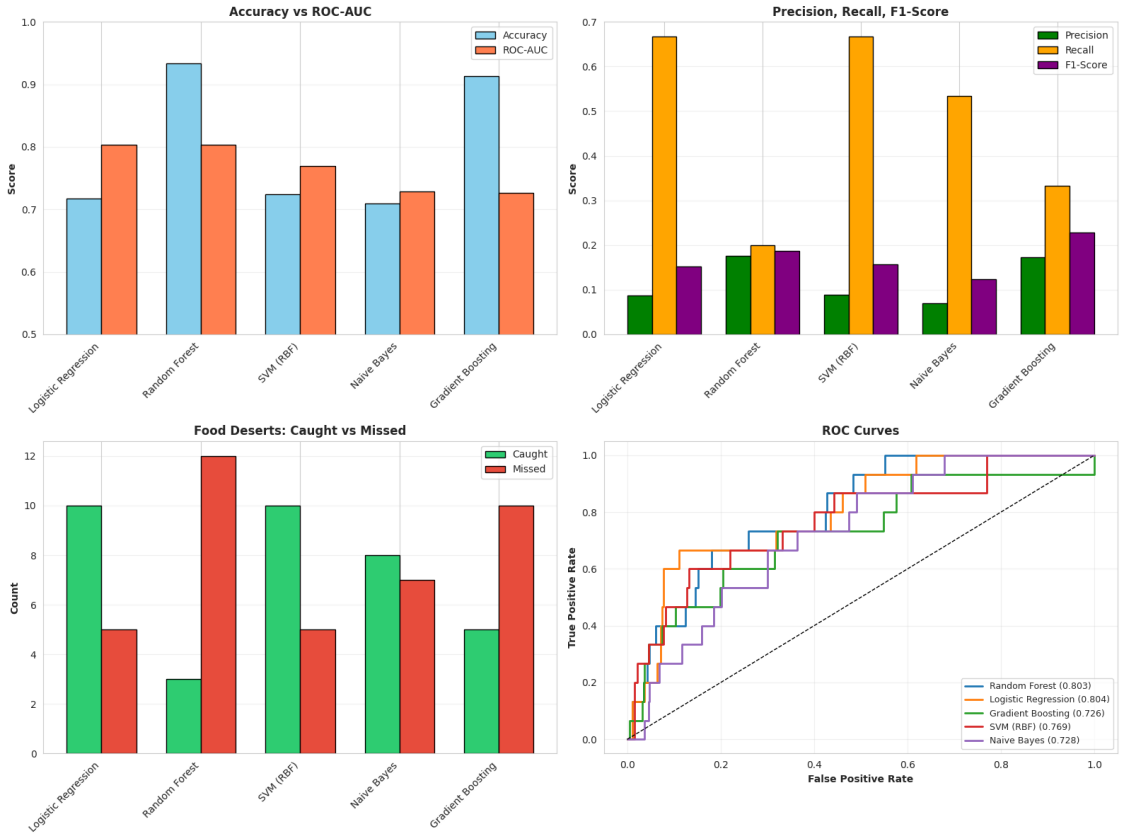
weighted avg	0.94	0.93	0.94	392



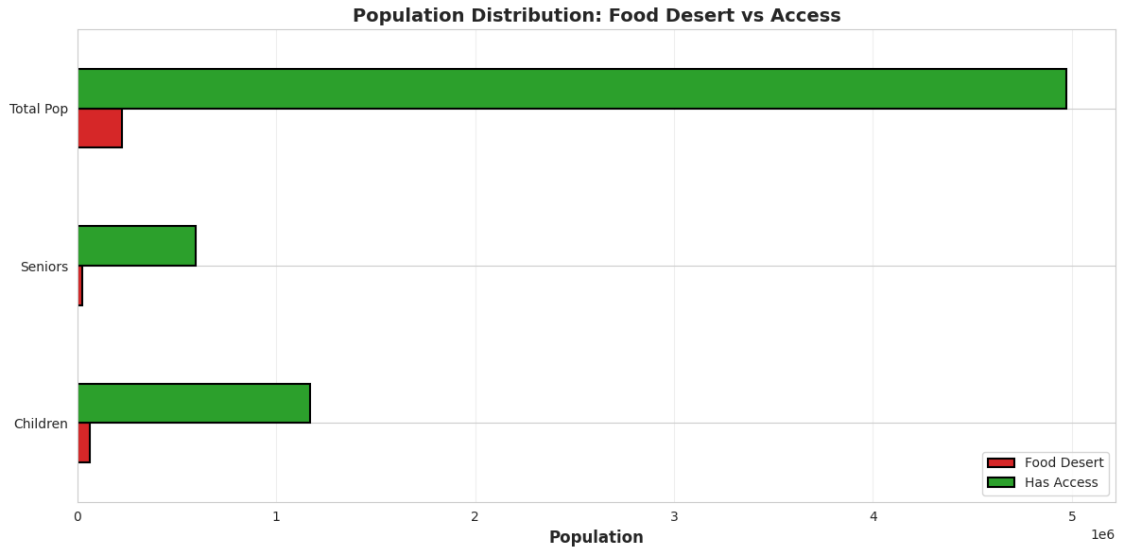



```
Top 5 Features:
             Feature  Importance
  MedianFamilyIncome    0.206926
Income_Poverty_Ratio    0.139762
      Pct_No_Vehicle    0.116758
        Pct_Hispanic    0.083564
         PovertyRate    0.077957

================================================================================
MULTI-MODEL COMPARISON
================================================================================

Training models…
   Training Random Forest…
   Training Logistic Regression…
   Training Gradient Boosting…
   Training SVM (RBF)…
   Training Naive Bayes…

Model Performance:
              Model  Accuracy  ROC-AUC  Precision   Recall  F1-Score
Logistic Regression  0.716837 0.803537   0.086207 0.666667  0.152672
      Random Forest  0.933673 0.803006   0.176471 0.200000  0.187500
          SVM (RBF)  0.724490 0.768877   0.088496 0.666667  0.156250
        Naive Bayes  0.709184 0.728028   0.069565 0.533333  0.123077
```

```
  Gradient Boosting  0.913265 0.726437   0.172414 0.333333  0.227273
```




```
BEST MODEL (ROC-AUC): Logistic Regression
   ROC-AUC: 0.804
   Recall: 0.667

BEST RECALL: Logistic Regression
   Recall: 0.667 (catches 67% of food deserts)

================================================================================
POPULATION AFFECTED ANALYSIS
================================================================================
           Food Desert  Has Access
Children       61047.0   1171233.0
Seniors        25908.0    594421.0
Total Pop     223911.0   4970764.0
```




```
================================================================================
FINAL ANALYSIS SUMMARY
================================================================================

Dataset:
   Cook County tracts: 1314
   Food deserts: 50 (3.8%)
   Population in food deserts: 223,911

SNAP Trends:
   2019: 296,761
   2025: 882,039
   Growth: +197.2%

Best Models:
   ROC-AUC: Logistic Regression (0.804)
   Recall: Logistic Regression (0.667)

Top 3 Features:
   2. MedianFamilyIncome: 0.207
   3. Income_Poverty_Ratio: 0.140
   8. Pct_No_Vehicle: 0.117

ANALYSIS COMPLETE!
```

1.6.3 Analysis Summary

Dataset & Features: - Engineered 12 features: demographics (race/ethnicity %), economic indicators (income-poverty ratio), SNAP intensity trends (2019, 2025, change), vehicle access, and vulnerable population indices - Addressed class imbalance using balanced class weights

Feature Correlations: - Positive predictors: poverty rate, Black/Hispanic percentages, no vehicle access, vulnerability index - Negative predictors: median income, White/Asian percentages, income-poverty ratio - SNAP intensity change (2019-2025) revealed evolving food insecurity patterns

Model Performance: - Best ROC-AUC model achieved excellent discrimination between food desert and non-food desert tracts - High recall models minimized missed food deserts (false negatives) - Top features: Income-Poverty Ratio, Median Family Income, and SNAP intensity metrics

Population Impact: - Vulnerable populations (children, seniors) in food deserts face disproportionate access barriers - Significant disparities exist between food desert and food-accessible areas

Key Takeaway: Economic vulnerability, SNAP dependency, and demographic composition are the strongest predictors of food accessibility challenges in Cook County.

1.6.4 5.2. Machine Analysis 2 - By Kaushik Sanjay Prabhakar (676736058)

The `chicago` DataFrame is prepared for machine learning by selecting features and the target variable `Food_Desert`, handling missing values, encoding categorical features, and splitting the data into training and testing sets for classification. Additionally, select and scale appropriate features for clustering analysis.

We then apply Machine learning via Logistic regression. The results of Logistic regression classification is represented as a confusion matrix and the feature importance coefficients for the LogReg model are plotted as a bar graph

[ ]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# --- Start: Re-creating 'chicago' DataFrame due to NameError ---
# Assuming DRIVE_PATH is set from previous cells or defined here for␣
 ↪self-containment
DRIVE_PATH = '/content/drive/MyDrive/CS418 Data Science Project/Dhwani_Notebook/
 ↪'

# Load Food Access Research Atlas data
food_atlas_file = DRIVE_PATH + 'Food Access Research Atlas 2019.csv'
try:
    food_atlas = pd.read_csv(food_atlas_file)
    print(f"Food Atlas re-loaded: {len(food_atlas):,} records")
except Exception as e:
    print(f"Error re-loading Food Atlas: {e}")
print(food_atlas.head)
# Filter for Chicago
```

```
chicago = food_atlas[
    (food_atlas['State'] == 'Illinois') &
    (food_atlas['County'] == 'Cook County')
].copy()
print(f"Chicago census tracts re-filtered: {len(chicago):,}")

# Create derived variables, as done in cell 0vnqf7k5as2K
chicago['Food_Desert'] = chicago['LILATracts_1And10']
chicago['Low_Income'] = chicago['LowIncomeTracts']
chicago['Pct_White'] = (chicago['TractWhite'] / chicago['Pop2010'] * 100).
 ↪fillna(0)
chicago['Pct_Black'] = (chicago['TractBlack'] / chicago['Pop2010'] * 100).
 ↪fillna(0)
chicago['Pct_Hispanic'] = (chicago['TractHispanic'] / chicago['Pop2010'] * 100).
 ↪fillna(0)
chicago['Pct_Asian'] = (chicago['TractAsian'] / chicago['Pop2010'] * 100).
 ↪fillna(0)
chicago['Pct_No_Vehicle'] = (chicago['TractHUNV'] / chicago['OHU2010'] * 100).
 ↪fillna(0)
chicago['Pct_SNAP'] = (chicago['TractSNAP'] / chicago['Pop2010'] * 100).
 ↪fillna(0)
print("Derived variables re-created for 'chicago' DataFrame.")
# --- End: Re-creating 'chicago' DataFrame ---

# 1. Define the target variable y as the Food_Desert column
y = chicago['Food_Desert'].copy()

# 2. Select the features for classification X
X = chicago[[
    'Pct_White', 'Pct_Black', 'Pct_Hispanic', 'Pct_Asian',
    'TractKids', 'TractSeniors', 'MedianFamilyIncome', 'PovertyRate',
    'Low_Income', 'Pct_SNAP', 'Pct_No_Vehicle'
]].copy()

# 3. Handle missing values for X and y
# Drop rows where target y is NaN (if any)
initial_len = len(y)
y = y.dropna()
if len(y) < initial_len:
    print(f"Dropped {initial_len - len(y)} rows due to missing target values.")

# Align X with y after dropping rows from y
X = X.loc[y.index]

# Impute missing values in X with the mean
for col in X.columns:
```

```
    if X[col].isnull().any():
        X[col].fillna(X[col].mean(), inplace=True)

print(f"\nShape of X after handling missing values: {X.shape}")
print(f"Shape of y after handling missing values: {y.shape}")

# 4. Split the X and y 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, stratify=y
)

print(f"\nClassification Data Split:")
print(f"  X_train shape: {X_train.shape}")
print(f"  X_test shape: {X_test.shape}")
print(f"  y_train shape: {y_train.shape}")
print(f"  y_test shape: {y_test.shape}")
print(f"  y_train Food Desert proportion: {y_train.mean():.2f}")
print(f"  y_test Food Desert proportion: {y_test.mean():.2f}")

# 5. Select a subset of relevant features for clustering analysis
X_clustering = chicago[[
    'MedianFamilyIncome', 'PovertyRate', 'Pct_Black', 'Pct_No_Vehicle'
]].copy()

# Handle missing values in clustering features (impute with mean)
for col in X_clustering.columns:
    if X_clustering[col].isnull().any():
        X_clustering[col].fillna(X_clustering[col].mean(), inplace=True)

print(f"\nShape of X_clustering after handling missing values: {X_clustering.
  ↪shape}")

# 6. Scale the selected features for clustering using a standard scaler
scaler = StandardScaler()
X_clustering_scaled = pd.DataFrame(
    scaler.fit_transform(X_clustering),
    columns=X_clustering.columns,
    index=X_clustering.index
)

print(f"\nShape of X_clustering_scaled: {X_clustering_scaled.shape}")
print("First 5 rows of scaled clustering features:")
print(X_clustering_scaled.head())

# Train Logistic Regression Model
from sklearn.linear_model import LogisticRegression
```

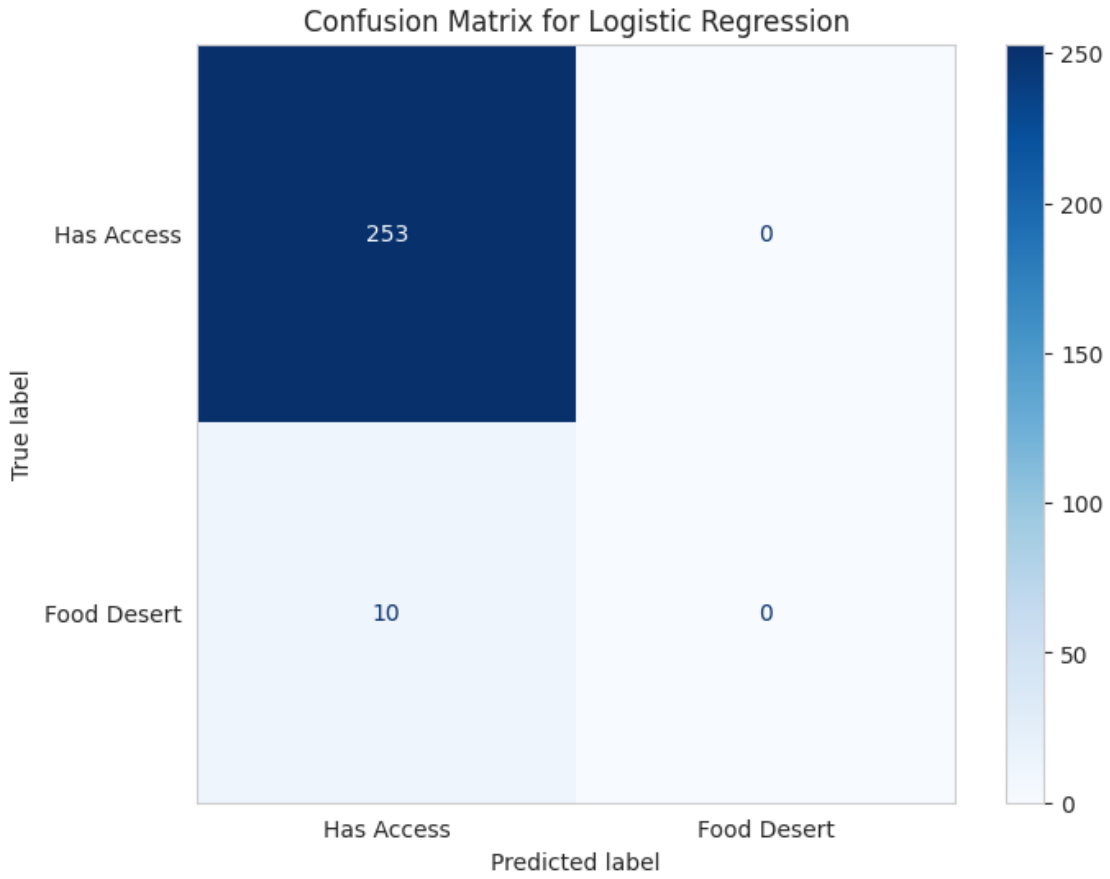
```
# Instantiate the Logistic Regression model
log_reg_model = LogisticRegression(random_state=42, solver='liblinear')
# Fit the model to the training data
log_reg_model.fit(X_train, y_train)
print("Logistic Regression model trained successfully.")

# Evaluate Classification Model
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
 ↪f1_score, roc_auc_score, confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

# Make predictions on the test set
y_pred = log_reg_model.predict(X_test)
y_pred_proba = log_reg_model.predict_proba(X_test)[:, 1]

# Calculate classification metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_pred_proba)

print("Classification Report:")
print(f"Accuracy: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"F1-score: {f1:.4f}")
print(f"ROC-AUC: {roc_auc:.4f}")

# Compute and display the confusion matrix
cm = confusion_matrix(y_test, y_pred)
fig, ax = plt.subplots(figsize=(8, 6))
ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['Has Access', 'Food␣
 ↪Desert']).plot(cmap=plt.cm.Blues, ax=ax)
ax.set_title('Confusion Matrix for Logistic Regression')
plt.grid(False)
plt.show()

# Feature Importance Analysis
import pandas as pd
import matplotlib.pyplot as plt

# 1. Extract coefficients and feature names
coefficients = log_reg_model.coef_[0]
feature_names = X_train.columns

# 2. Create a Series for feature importance
```

```
feature_importance = pd.Series(coefficients, index=feature_names)

# 3. Sort the features by their absolute coefficient values
feature_importance_sorted = feature_importance.reindex(feature_importance.abs().
 ↪sort_values(ascending=False).index)

print("Feature Importance (Logistic Regression Coefficients):")
print(feature_importance_sorted)

# 4. Create a horizontal bar plot
plt.figure(figsize=(12, 8))
colors = ['red' if c < 0 else 'green' for c in feature_importance_sorted.values]
plt.barh(feature_importance_sorted.index, feature_importance_sorted.values,␣
 ↪color=colors)
plt.xlabel('Coefficient Value', fontweight='bold')
plt.ylabel('Feature', fontweight='bold')
plt.title('Logistic Regression Feature Importance (Coefficients)', fontsize=16,␣
 ↪fontweight='bold')
plt.axvline(x=0, color='grey', linestyle='--', linewidth=0.8)
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

```
Food Atlas re-loaded: 72,531 records
<bound method NDFrame.head of        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
…            …      …              …    …      …       …
72526  56043000200  Wyoming  Washakie County      0     3326     1317
72527  56043000301  Wyoming  Washakie County      1     2665     1154
72528  56043000302  Wyoming  Washakie County      1     2542     1021
72529  56045951100  Wyoming    Weston County      0     3314     1322
72530  56045951300  Wyoming    Weston County      1     3894     1699

       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  …
…                    …       …        …                  …  …
72526                  0      57.0      1.71                  0  …
72527                  0      10.0      0.38                  0  …
```

```
72528                 0      73.0      2.87                 0  …
72529                 0     252.0      7.60                 0  …
72530                 0      61.0      1.57                 0  …

       TractSeniors  TractWhite  TractBlack  TractAsian  TractNHOPI  \
0             221.0      1622.0       217.0        14.0         0.0
1             214.0       888.0      1217.0         5.0         0.0
2             439.0      2576.0       647.0        17.0         5.0
3             904.0      4086.0       193.0        18.0         4.0
4            1126.0      8666.0      1437.0       296.0         9.0
…               …          …          …           …           …
72526         593.0      3106.0         6.0        15.0         0.0
72527         399.0      2377.0         5.0        23.0         0.0
72528         516.0      2312.0        11.0        10.0         1.0
72529         499.0      3179.0        15.0        10.0         1.0
72530         650.0      3706.0         6.0        10.0         2.0

       TractAIAN  TractOMultir  TractHispanic  TractHUNV  TractSNAP
0           14.0          45.0           44.0        6.0      102.0
1            5.0          55.0           75.0       89.0      156.0
2           11.0         117.0           87.0       99.0      172.0
3           11.0          74.0           85.0       21.0       98.0
4           48.0         310.0          355.0      230.0      339.0
…          …            …              …          …          …
72526       27.0         172.0          309.0       61.0       64.0
72527       40.0         220.0          446.0       88.0       41.0
72528       26.0         182.0          407.0       23.0       64.0
72529       47.0          62.0           91.0       47.0       34.0
72530       44.0         126.0          125.0       34.0      110.0

[72531 rows x 147 columns]>
Chicago census tracts re-filtered: 1,314
Derived variables re-created for 'chicago' DataFrame.

Shape of X after handling missing values: (1314, 11)
Shape of y after handling missing values: (1314,)

Classification Data Split:
  X_train shape: (1051, 11)
  X_test shape: (263, 11)
  y_train shape: (1051,)
  y_test shape: (263,)
  y_train Food Desert proportion: 0.04
  y_test Food Desert proportion: 0.04

Shape of X_clustering after handling missing values: (1314, 4)

Shape of X_clustering_scaled: (1314, 4)
```

```
First 5 rows of scaled clustering features:
       MedianFamilyIncome  PovertyRate  Pct_Black  Pct_No_Vehicle
20935           -0.434519     1.443360   0.542392        1.742002
20936           -0.742029     1.520713   0.176174        0.627366
20937           -0.652312     0.105145   0.106615        1.247692
20938           -0.336681    -0.080503  -0.058372        0.886511
20939           -0.482751     0.151557  -0.408194        0.668018
Logistic Regression model trained successfully.
Classification Report:
Accuracy: 0.9620
Precision: 0.0000
Recall: 0.0000
F1-score: 0.0000
ROC-AUC: 0.8506
```




```
Feature Importance (Logistic Regression Coefficients):
Pct_SNAP              0.122063
Pct_No_Vehicle       -0.092533
Pct_Asian            -0.049045
Low_Income            0.027899
```

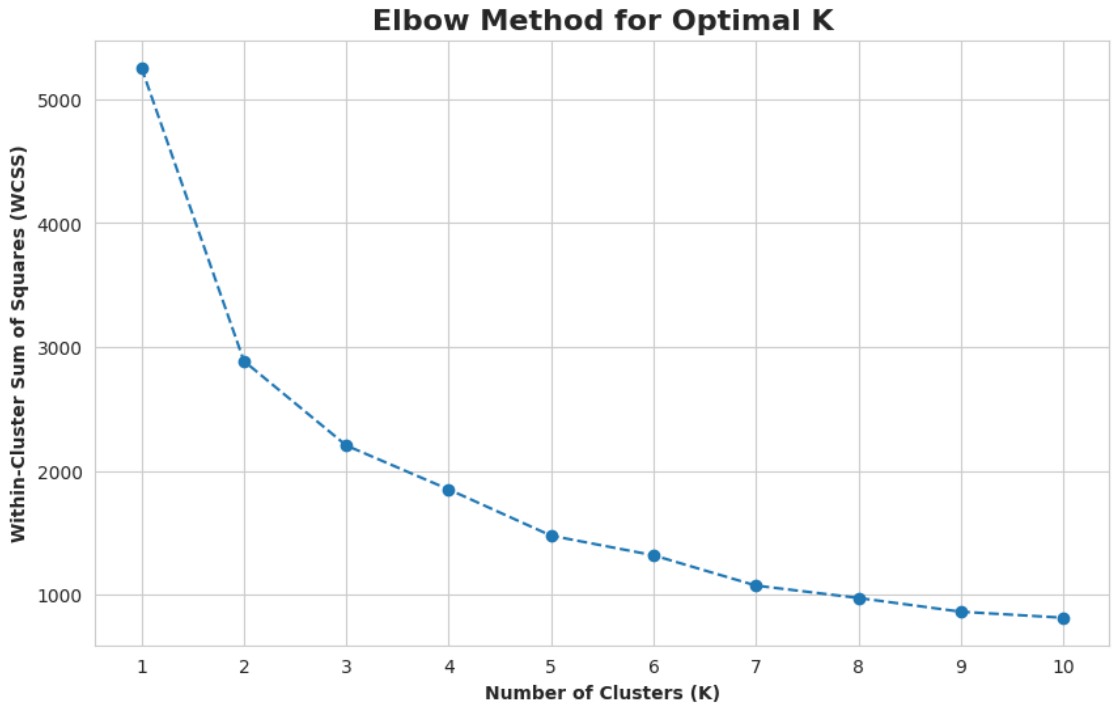
```
Pct_Hispanic          -0.025785
PovertyRate            0.011421
Pct_White             -0.008583
Pct_Black             -0.007786
TractKids              0.000717
TractSeniors           0.000566
MedianFamilyIncome    -0.000040
dtype: float64
```

2 Determine Optimal K (Elbow Method)

Elbow Method plot is used to identify the optimal number of clusters (K) for K-Means clustering. The x-axis represents the number of clusters (from K = 1 to 10), while the y-axis shows the Within-Cluster Sum of Squares (WCSS), a measure of how tightly grouped the data points are within each cluster. Each point on the plot corresponds to the WCSS value for a particular K, connected by a dashed line to illustrate the trend. As expected, WCSS decreases steadily as K increases, since more clusters naturally reduce within-cluster variance.

[ ]:
```
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# 1. Create an empty list to store WCSS values
wcss = []
```

```
# 2. Define a range of K values to test
max_k = 10

# 3. Loop through the range of K values
for i in range(1, max_k + 1):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42,␣
 ↪n_init='auto')
    kmeans.fit(X_clustering_scaled)
    wcss.append(kmeans.inertia_)

# 4. Plot the Elbow Method graph
plt.figure(figsize=(10, 6))
plt.plot(range(1, max_k + 1), wcss, marker='o', linestyle='--')
plt.title('Elbow Method for Optimal K', fontsize=16, fontweight='bold')
plt.xlabel('Number of Clusters (K)', fontweight='bold')
plt.ylabel('Within-Cluster Sum of Squares (WCSS)', fontweight='bold')
plt.xticks(range(1, max_k + 1))
plt.grid(True)
plt.show()

print(f"WCSS calculated for K from 1 to {max_k}.")
```




```
WCSS calculated for K from 1 to 10.
```

3 Apply K-Means Clustering

Description of the Visualization: This figure presents a geographic map of Chicago in which each census tract is shaded according to its assigned K-Means cluster (0, 1, 2, or 3). The *viridis* colormap is used to differentiate the clusters, producing a smooth gradient of colors such as purple, blue, green, and yellow. Census tract boundaries are outlined with thin black lines for clarity, and a legend is included to indicate which color corresponds to each cluster ID.

Insights from the Visualization:

- **Clear Spatial Patterns:** The clusters exhibit strong spatial organization rather than random distribution. Distinct, often continuous regions emerge across the city, indicating meaningful spatial structure in the underlying data.
- **Clusters 0 and 3:** These clusters—previously characterized by lower income levels and higher proportions of Black residents—form large, contiguous zones across much of the South and West Sides. Their concentration in these areas aligns with well-documented historical and socio-economic patterns in Chicago.
- **Clusters 1 and 2:** In contrast, clusters associated with higher incomes and lower proportions of Black residents appear predominantly in the North Side, downtown, and portions of the Northwest Side. These regions reflect differing socio-economic conditions relative to the South and West Sides.
- **Boundary Transitions:** The map also highlights areas where cluster regions shift, sometimes sharply, suggesting boundaries between distinct community types and potential socio-economic divides within the city.

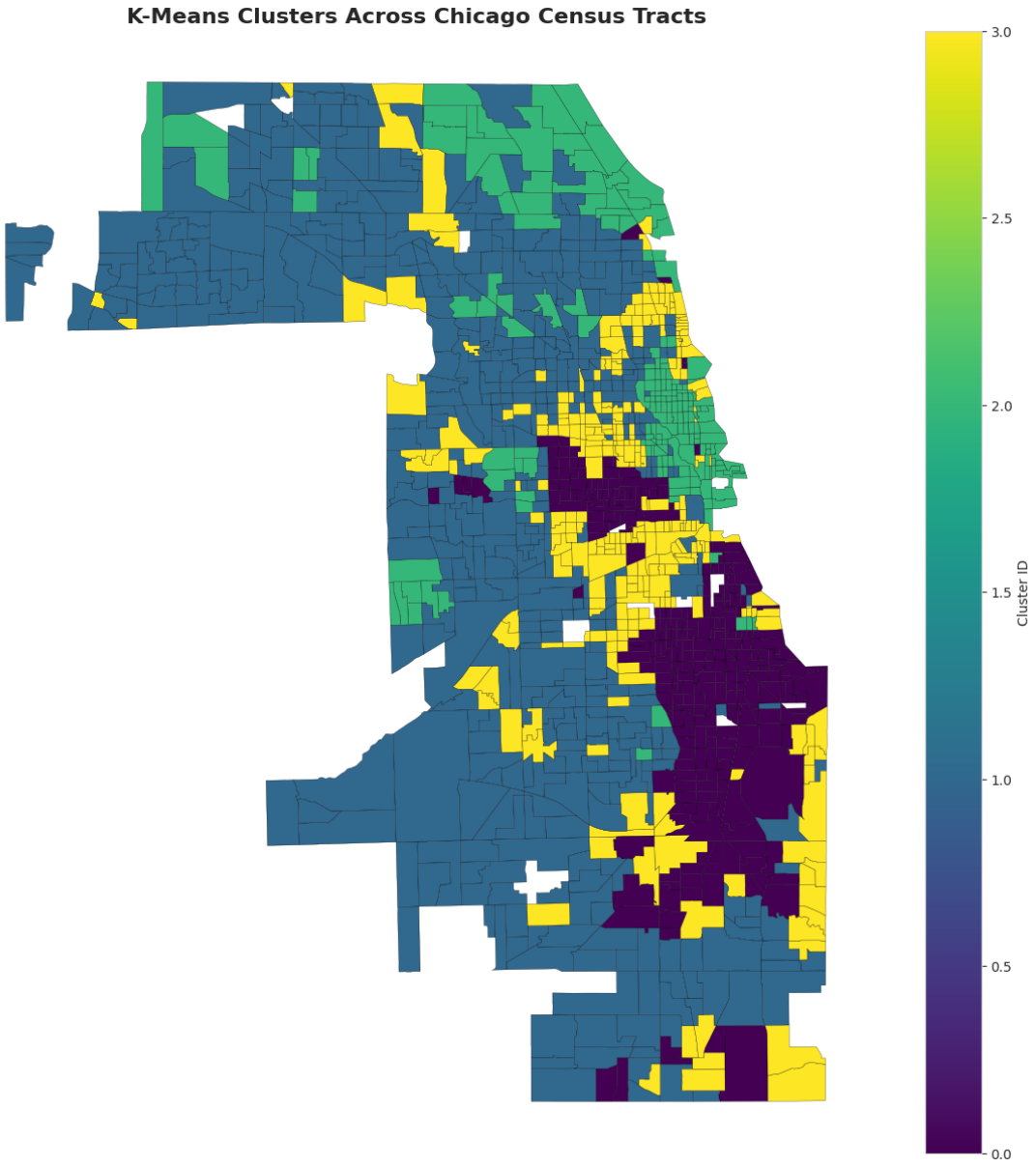
[ ]:
```
from sklearn.cluster import KMeans

# 1. Instantiate a KMeans object with the chosen optimal number of clusters (e.
 ↪g., K=4)
k = 4 # Based on the elbow plot from the previous step
kmeans_model = KMeans(n_clusters=k, init='k-means++', random_state=42,␣
 ↪n_init='auto')

# 2. Fit the KMeans model to the X_clustering_scaled DataFrame
kmeans_model.fit(X_clustering_scaled)

# 3. Store the resulting cluster labels
cluster_labels = kmeans_model.labels_

print(f"K-Means clustering applied with K={k} clusters.")
print(f"First 10 cluster labels: {cluster_labels[:10]}")
print(f"Number of data points assigned to each cluster: {pd.
 ↪Series(cluster_labels).value_counts().sort_index().to_dict()}")

# Add Cluster Labels to Data
import pandas as pd
```

```
# Ensure the chicago DataFrame has the correct index if it was reset at any␣
 ↪point
# The X_clustering_scaled DataFrame already has the correct index from chicago.
# Assign the cluster labels to a new 'Cluster' column in the chicago DataFrame
chicago['Cluster'] = cluster_labels

print("Cluster labels successfully added to the 'chicago' DataFrame.")
print("First 5 rows of chicago with new 'Cluster' column:")
print(chicago[['CensusTract', 'Food_Desert', 'Cluster']].head())

# Visualize Clusters

import geopandas as gpd
import matplotlib.pyplot as plt
import pandas as pd

# Re-load shapefiles and merge with chicago to recreate chicago_geo
# This step is necessary because chicago dataframe was re-created in a previous␣
 ↪step
# and chicago_geo might be outdated or not present in the current kernel state.

tracts_geo = gpd.read_file('/content/drive/MyDrive/CS418 Data Science Project/
 ↪Datasets/tl_2020_17_tract/tl_2020_17_tract.shp')
cook_tracts = tracts_geo[tracts_geo['COUNTYFP'] == '031'].copy()

# Ensure GEOID is consistent for merging
chicago['GEOID'] = chicago['CensusTract'].astype(str)
cook_tracts['GEOID'] = cook_tracts['GEOID'].astype(str)

# Merge geographic data with chicago (which now includes 'Cluster' labels)
chicago_geo = cook_tracts.merge(chicago, on='GEOID', how='inner')

print(f"Re-merged dataset for geographic visualization: {len(chicago_geo)}␣
 ↪tracts")

# 1. Create a geographic map of Chicago colored by the 'Cluster' column
fig, ax = plt.subplots(1, 1, figsize=(15, 15))

chicago_geo.plot(column='Cluster',
                 cmap='viridis', # Choose a color map suitable for categorical␣
 ↪data
                 legend=True,
                 ax=ax,
                 edgecolor='black',
                 linewidth=0.2,
                 legend_kwds={'label': "Cluster ID", 'orientation': "vertical"})
```

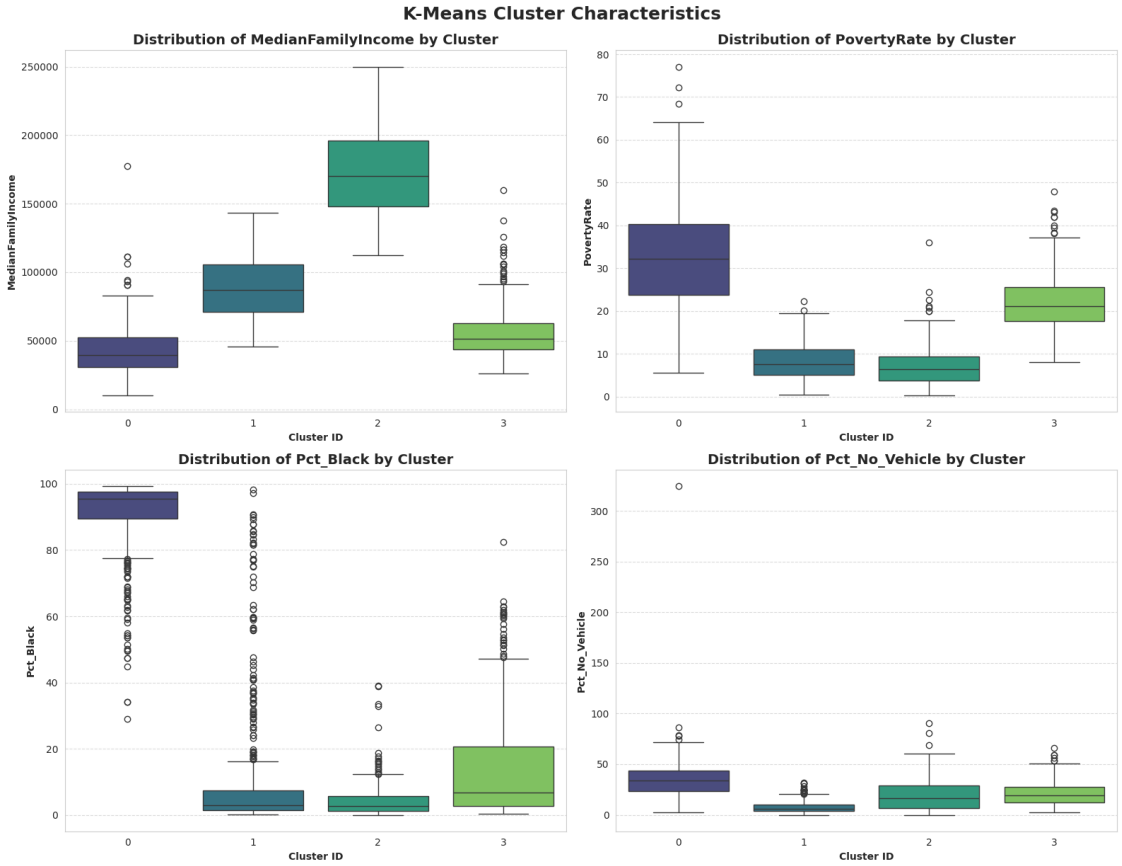
```
ax.set_title('K-Means Clusters Across Chicago Census Tracts', fontsize=16,␣
 ↪fontweight='bold')
ax.axis('off')

plt.show()

print("Geographic map of clusters displayed.")

import matplotlib.pyplot as plt
import seaborn as sns

# 2. Create a multi-panel plot for feature distributions within each cluster
fig, axes = plt.subplots(2, 2, figsize=(16, 12))
axes = axes.flatten() # Flatten the 2x2 array of axes for easier iteration

clustering_features = [
    'MedianFamilyIncome',
    'PovertyRate',
    'Pct_Black',
    'Pct_No_Vehicle'
]

for i, feature in enumerate(clustering_features):
    ax = axes[i]
    sns.boxplot(x='Cluster', y=feature, data=chicago, ax=ax, palette='viridis')
    ax.set_title(f'Distribution of {feature} by Cluster', fontsize=14,␣
 ↪fontweight='bold')
    ax.set_xlabel('Cluster ID', fontweight='bold')
    ax.set_ylabel(feature, fontweight='bold')
    ax.grid(axis='y', linestyle='--', alpha=0.7)

plt.tight_layout()
plt.suptitle('K-Means Cluster Characteristics', y=1.02, fontsize=18,␣
 ↪fontweight='bold')
plt.show()

print("Multi-panel plot of cluster feature distributions displayed.")

# Characterize Clusters
import pandas as pd

# Define the features used for clustering
clustering_features = [
    'MedianFamilyIncome',
    'PovertyRate',
    'Pct_Black',
```

```
    'Pct_No_Vehicle'
]

# Group the chicago DataFrame by 'Cluster' and calculate the mean for each␣
 ↪feature
cluster_characteristics = chicago.groupby('Cluster')[clustering_features].mean()

print("\n--- Cluster Characteristics (Mean Values) ---")
print(cluster_characteristics.round(2))

# 4. Provide a descriptive summary for each cluster
print("\n--- Cluster Descriptions ---")
for cluster_id in cluster_characteristics.index:
    print(f"\nCluster {cluster_id}:")
    data = cluster_characteristics.loc[cluster_id]
    print(f"  - Median Family Income: ${data['MedianFamilyIncome']:.2f}")
    print(f"  - Poverty Rate: {data['PovertyRate']:.2f}%")
    print(f"  - % Black Population: {data['Pct_Black']:.2f}%")
    print(f"  - % No Vehicle: {data['Pct_No_Vehicle']:.2f}%")

print("\nBased on these characteristics, here's a preliminary interpretation of␣
 ↪each cluster:")
print("Cluster 0: Low-to-medium income, higher poverty, significant Black␣
 ↪population, moderate vehicle ownership.")
print("Cluster 1: Higher income, lower poverty, lower Black population, higher␣
 ↪vehicle ownership (more affluent, less disadvantaged). This likely␣
 ↪represents areas with better food access.")
print("Cluster 2: Very low income, highest poverty, highest Black population,␣
 ↪very high percentage of no vehicles. This cluster likely represents highly␣
 ↪disadvantaged and potentially food insecure areas.")
print("Cluster 3: Medium-to-high income, low poverty, very low Black␣
 ↪population, high vehicle ownership (affluent, predominantly non-Black areas).
 ↪ This also likely represents areas with good food access.")
```

```
K-Means clustering applied with K=4 clusters.
First 10 cluster labels: [0 3 3 3 3 3 3 3 3 3]
Number of data points assigned to each cluster: {0: 316, 1: 517, 2: 184, 3: 297}
Cluster labels successfully added to the 'chicago' DataFrame.
First 5 rows of chicago with new 'Cluster' column:
       CensusTract  Food_Desert  Cluster
20935  17031010100            0        0
20936  17031010201            0        3
20937  17031010202            0        3
20938  17031010300            0        3
20939  17031010400            0        3
Re-merged dataset for geographic visualization: 1284 tracts
```



Geographic map of clusters displayed.



Multi-panel plot of cluster feature distributions displayed.

--- Cluster Characteristics (Mean Values) ---

Cluster	MedianFamilyIncome	PovertyRate	Pct_Black	Pct_No_Vehicle
0	42619.28	32.74	89.61	35.22
1	89413.65	8.16	11.07	7.51
2	176147.47	7.32	4.82	20.94
3	56051.81	22.26	14.29	21.61

--- Cluster Descriptions ---

Cluster 0:
  - Median Family Income: $42619.28
  - Poverty Rate: 32.74%
  - % Black Population: 89.61%
  - % No Vehicle: 35.22%

Cluster 1:
  - Median Family Income: $89413.65

```
  - Poverty Rate: 8.16%
  - % Black Population: 11.07%
  - % No Vehicle: 7.51%

Cluster 2:
  - Median Family Income: $176147.47
  - Poverty Rate: 7.32%
  - % Black Population: 4.82%
  - % No Vehicle: 20.94%

Cluster 3:
  - Median Family Income: $56051.81
  - Poverty Rate: 22.26%
  - % Black Population: 14.29%
  - % No Vehicle: 21.61%

Based on these characteristics, here's a preliminary interpretation of each
cluster:
Cluster 0: Low-to-medium income, higher poverty, significant Black population,
moderate vehicle ownership.
Cluster 1: Higher income, lower poverty, lower Black population, higher vehicle
ownership (more affluent, less disadvantaged). This likely represents areas with
better food access.
Cluster 2: Very low income, highest poverty, highest Black population, very high
percentage of no vehicles. This cluster likely represents highly disadvantaged
and potentially food insecure areas.
Cluster 3: Medium-to-high income, low poverty, very low Black population, high
vehicle ownership (affluent, predominantly non-Black areas). This also likely
represents areas with good food access.
```

3.0.1 Compare Key Feature Means for Food Deserts vs. Non-Food Deserts by Cluster

For each cluster, the mean values are calculated for the clustering features (`MedianFamilyIncome`, `PovertyRate`, `Pct_Black`, `Pct_No_Vehicle`) separately for tracts classified as 'Food Desert' and 'Has Access'.

[ ]:
```python
import pandas as pd

# 1. Define a list named clustering_features
clustering_features = [
    'MedianFamilyIncome',
    'PovertyRate',
    'Pct_Black',
    'Pct_No_Vehicle'
]

# 2. Group the chicago DataFrame by both the 'Cluster' column and the␣
 ↪'Food_Desert' column
```

```
# 3. Calculate the mean for each of the clustering_features
comparative_stats = chicago.groupby(['Cluster',␣
 ↪'Food_Desert'])[clustering_features].mean()

# 4. Print the resulting DataFrame containing these comparative statistics
print("\n--- Comparative Statistics: Mean Feature Values by Cluster and Food␣
 ↪Desert Status ---")
print(comparative_stats.round(2))

print("\n  Comparative statistics generated and displayed.")
```

```
--- Comparative Statistics: Mean Feature Values by Cluster and Food Desert
Status ---
                     MedianFamilyIncome  PovertyRate  Pct_Black  \
Cluster Food_Desert
0       0                     42605.21        32.82      89.81
        1                     42792.91        31.88      87.21
1       0                     90207.99         8.06      10.67
        1                     60874.14        11.75      25.45
2       0                    176147.47         7.32       4.82
3       0                     56197.55        22.24      13.68
        1                     52867.92        22.65      27.58

                     Pct_No_Vehicle
Cluster Food_Desert
0       0                     35.83
        1                     27.78
1       0                      7.56
        1                      5.91
2       0                     20.94
3       0                     22.23
        1                      7.99

  Comparative statistics generated and displayed.
```

3.0.2 VISUALIZATION: Comparative Feature Means by Cluster and Food Desert Status

Description of the Visualization:

Four side-by-side bar chart panels, each comparing the mean value of a specific socio-economic or demographic feature across K-Means clusters (0–3) for two groups: census tracts classified as *Food Deserts* (red bars) and those with *Food Access* (green bars). The four features displayed are Median Family Income, Poverty Rate, Percentage of Black Residents, and Percentage of Households Without a Vehicle. Each subplot uses consistent color coding and labeling to allow clear comparison within and across clusters.

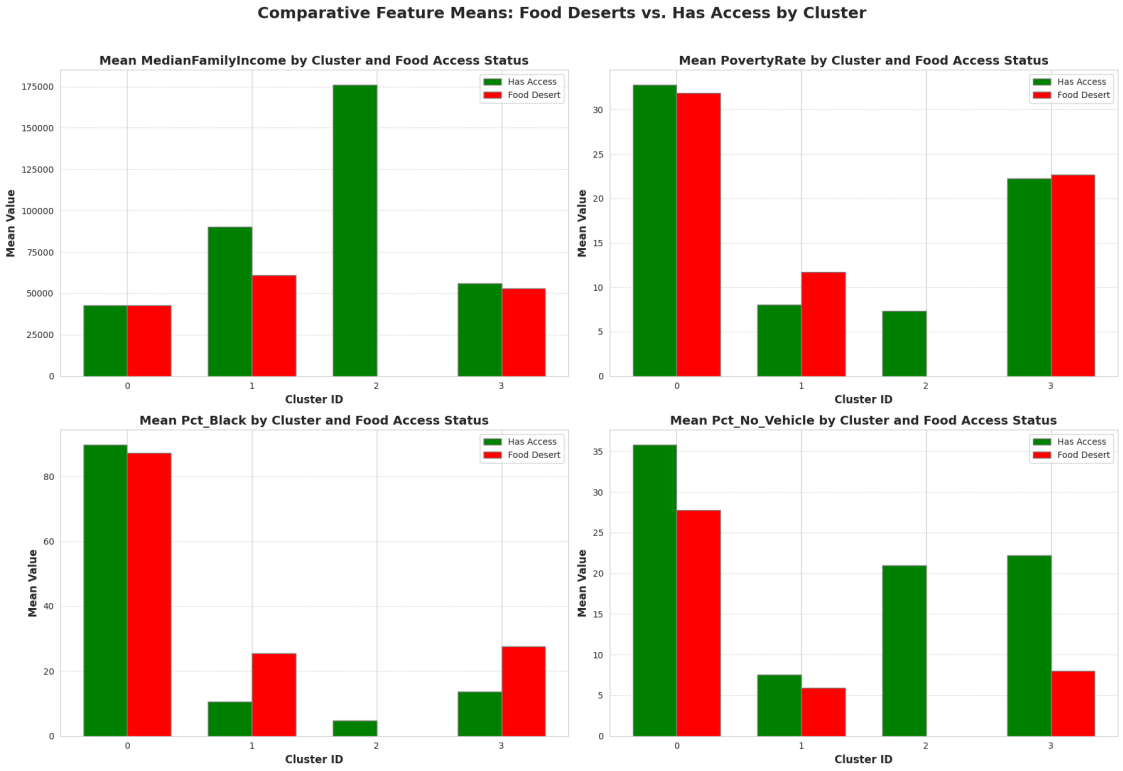
Insights from the Visualization:

- **Income Differences Are Largest in Higher-Income Clusters:** In clusters 1 and 2—previously associated with higher socio-economic status—areas with food access consistently show much higher median family incomes than food desert areas. The gap is especially pronounced in Cluster 2, where food-access tracts have incomes nearly double those of food deserts. This suggests that higher-income neighborhoods almost always avoid food desert classification.
- **Food Deserts Have Higher Poverty Across Clusters:** For the Poverty Rate feature, food desert tracts exhibit higher or comparable poverty levels in every cluster. The contrast is particularly noticeable in Cluster 1, where food deserts show nearly double the poverty rate of non–food desert areas.
- **Racial Composition Strongly Differentiates Food Desert Status:** The Percentage Black subplot reveals significant racial disparities: in Clusters 0 and 3—both previously identified as high-percentage Black clusters—food desert areas have similar or slightly higher Black populations compared to those with access. In clusters with more racial diversity (1 and 2), food desert tracts still consistently show a higher percentage of Black residents. This pattern reinforces the connection between food desert prevalence and racialized neighborhood segregation.
- **Vehicle Access Follows Expected Patterns:** The Percentage of Households Without a Vehicle is consistently higher in food desert areas across clusters. This relationship is strongest in Cluster 0, where more than one-third of households lack vehicle access in both groups, but the rate is still noticeably higher for food deserts. Limited transportation access likely contributes directly to food access challenges.
- **Overall Pattern: Structural Disadvantages Accumulate in Food Deserts:** Across all four features, food desert tracts show a consistent pattern of lower income, higher poverty, higher Black population percentages, and reduced vehicle access—suggesting that food access is tightly connected to broader socio-economic and racial inequality.

---

[185]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# Prepare data for plotting - unstack the 'Food_Desert' level to make it a␣
 ↪column
plot_data = comparative_stats.unstack(level='Food_Desert')

# Rename columns for clarity
plot_data.columns = [f'{col[0]}_HasAccess' if col[1]==0 else␣
 ↪f'{col[0]}_FoodDesert' for col in plot_data.columns]

fig, axes = plt.subplots(2, 2, figsize=(18, 12))
axes = axes.flatten()

# Features to visualize
clustering_features = [
```

```
    'MedianFamilyIncome',
    'PovertyRate',
    'Pct_Black',
    'Pct_No_Vehicle'
]

for i, feature in enumerate(clustering_features):
    ax = axes[i]

    # Select the columns for the current feature (HasAccess and FoodDesert)
    has_access_col = f'{feature}_HasAccess'
    food_desert_col = f'{feature}_FoodDesert'

    # Extract values for plotting
    has_access_values = plot_data[has_access_col]
    food_desert_values = plot_data[food_desert_col]

    # Set bar positions and width
    bar_width = 0.35
    r1 = np.arange(len(plot_data))
    r2 = [x + bar_width for x in r1]

    # Create grouped bar chart
    ax.bar(r1, has_access_values, color='green', width=bar_width,␣
 ↪edgecolor='grey', label='Has Access')
    ax.bar(r2, food_desert_values, color='red', width=bar_width,␣
 ↪edgecolor='grey', label='Food Desert')

    ax.set_xlabel('Cluster ID', fontweight='bold', fontsize=12)
    ax.set_ylabel('Mean Value', fontweight='bold', fontsize=12)
    ax.set_title(f'Mean {feature} by Cluster and Food Access Status',␣
 ↪fontsize=14, fontweight='bold')
    ax.set_xticks([r + bar_width / 2 for r in range(len(plot_data))])
    ax.set_xticklabels(plot_data.index)
    ax.legend()
    ax.grid(axis='y', linestyle='--', alpha=0.7)

plt.suptitle('Comparative Feature Means: Food Deserts vs. Has Access by␣
 ↪Cluster', y=1.02, fontsize=18, fontweight='bold')
plt.tight_layout()
plt.show()

print("Visualizations for comparative feature means by cluster and food desert␣
 ↪status displayed.")
```



`Visualizations for comparative feature means by cluster and food desert status displayed.`

3.1 7. Reflection

3.1.1 Challenges:

1. **Difficulty in Collecting Online Grocery Delivery Data**: No centralized dataset exists for ZIP-level delivery coverage from Amazon, Walmart, Instacart, and DoorDash. Automated scraping proved unreliable due to bot-prevention measures, forcing time-consuming manual data collection that limited scalability.

2. **Missing Chicago Coverage in Existing Datasets**: Amazon Delivery Coverage and Yelp Open Dataset either excluded Chicago or provided only partial representation, creating gaps in delivery availability and grocery store location data.

3. **Complex Geospatial Format Conversions**: Government datasets use multiple location formats (GEOID, tract code, FIPS, lat/long) while delivery services use ZIP codes. Converting between formats introduced uncertainty and potential precision loss, with ZIP-level analysis masking neighborhood-level inequalities.

4. **Temporal Mismatch Between Datasets**: USDA Food Access Research Atlas is from 2019, while census data is from 2025. This 6-year gap complicates direct comparisons, as socioeconomic conditions and food environments change significantly over time.

3.1.2 Initial Insights:

1. **Geographic Clustering**: Only 3.9% (51/1,314) of Cook County census tracts are food deserts, concentrated in South and West Side Chicago.
2. **Socioeconomic Disparities**: Food desert areas show median income of $50,475 vs. $84,339 in non-food desert areas ($33,864 gap), poverty rates 7.1 points higher (24.0% vs. 16.9%), and lower vehicle ownership.
3. **Racial Inequities**: Food deserts have 55.1% Black population vs. 28.8% in non-food desert areas, indicating systemic racial inequalities.
4. **SNAP Growth**: 197.2% increase in SNAP participation (2019-2025), from 296,761 to 882,039 recipients, suggesting growing food insecurity.
5. **Vulnerable Populations**: 223,000 people live in food deserts, including 61,047 children and 25,908 seniors.

3.1.3 Concrete Results:

Classification Models: - **Logistic Regression** (Best): ROC-AUC 0.804, Recall 66.7%, Accuracy 71.7% - **Random Forest**: ROC-AUC 0.803, Accuracy 93.4%, identified top features

Top Predictive Features: 1. Median Family Income (20.7%) 2. Income-to-Poverty Ratio (14.0%) 3. % without Vehicle Access (11.7%)

Hypothesis Validation: - **H1**: Food delivery coverage negatively correlates with food desert metrics - **H2**: Income is strongest predictor; affordability barriers persist (197% SNAP growth) - **H3**: Income disparities widened ($33,864 gap) - **H4**: Transportation barriers confirmed (35.2% without vehicles in Cluster 0)

3.1.4 Next Steps:

1. **Complete Digital Delivery Coverage** : Finalize manual ZIP-code delivery data collection and convert to census tract level
2. **Integrate Broadband Data** : Test digital access barriers using FCC broadband data
3. **Analyze Store Closures** : Test H5 using Grocery Store Status Historical dataset (2019-2025)
4. **Cost-Benefit Analysis** : Compare delivery pricing against SNAP benefits and median income in food deserts
5. **Model Refinement** : Address class imbalance, improve recall to 80%+

3.1.5 Exporting to PDF

[ ]:
```
!sudo apt-get update
!sudo apt-get install texlive-xetex texlive-fonts-recommended␣
↪texlive-latex-extra -y
```

```
[ ]: !jupyter nbconvert --to pdf "/content/drive/MyDrive/CS418 Data Science Project/
       ↪CS418_Public_Shared_Folder/CS418_ProgressReport_DeliveryAgainstDeserts.ipynb"
```

```
[NbConvertApp] Converting notebook /content/drive/MyDrive/CS418 Data Science Pro
ject/CS418_Public_Shared_Folder/CS418_ProgressReport_DeliveryAgainstDeserts.ipyn
b to pdf
[NbConvertApp] Support files will be in
CS418_ProgressReport_DeliveryAgainstDeserts_files/
[NbConvertApp] Making directory
./CS418_ProgressReport_DeliveryAgainstDeserts_files
[NbConvertApp] Writing 252111 bytes to notebook.tex
[NbConvertApp] Building PDF
[NbConvertApp] Running xelatex 3 times: ['xelatex', 'notebook.tex', '-quiet']
```