DATA_698_MASTERS_PROJECT

May 8, 2025

1 Necessary Packages

[1]:
```
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow import keras
from keras.src.models import Sequential
from keras.src.layers import Dense, LSTM, Dropout
from keras.src.regularizers import L2
from keras.src.callbacks import EarlyStopping, ModelCheckpoint
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
```

2 LSTM Experiment 1

- This LSTM only imputes the data, it does nothing else to infuence the data or the model. In other words these are fairly standard parameters for and LSTM model.

[ ]:
```
filepath = "C:\\Users\\jashb\\OneDrive\\Documents\\Masters Data Science\\Spring␣
↪2025\\DATA 698\\Masters Project\\final_data.csv"
df = pd.read_csv(filepath)
print(df.head())

numeric_columns = df.select_dtypes(include=['float64', 'int64']).columns
df[numeric_columns] = df[numeric_columns].fillna(df[numeric_columns].mean())

# Impute categorical columns with the most frequent value
categorical_columns = df.select_dtypes(include=['object']).columns
df[categorical_columns] = df[categorical_columns].
↪fillna(df[categorical_columns].mode().iloc[0])


# Step: Drop rows with missing target
df = df.dropna(subset=['percent_food_insecure'])
print(df.head())
# Step 3: Drop rows with missing target
```

```python
df = df.dropna(subset=['percent_food_insecure'])

print(df.head())

# Step 4: Convert 'rural_urban' to numeric
df['rural_urban'] = pd.factorize(df['rural_urban'])[0]

print(df[['rural_urban']].head())

# Step 5: Create lag features
df = df.sort_values(['fips', 'year'])
df['food_insecure_lag1'] = df.groupby('fips')['percent_food_insecure'].shift(1)
df['food_insecure_lag2'] = df.groupby('fips')['percent_food_insecure'].shift(2)

print(df[['fips', 'year', 'percent_food_insecure', 'food_insecure_lag1',␣
 ↪'food_insecure_lag2']].head())

# Step 6: Drop rows with missing lag features
df = df.dropna(subset=['food_insecure_lag1', 'food_insecure_lag2'])

print(df.head())

# Step 7: Select features
features = [
    'percent_household_income_required_for_child_care_expenses',
    'food_environment_index',
    'percent_fair_or_poor_health',
    'percent_unemployed',
    'percent_children_in_poverty',
    'percent_severe_housing_problems',
    'percent_completed_high_school',
    'percent_frequent_mental_distress',
    'percent_uninsured_children',
    'percent_disconnected_youth',
    'spending_per_pupil',
    'school_funding_adequacy',
    'high_school_graduation_rate',
    'median_household_income',
    'gender_pay_gap',
    'percent_enrolled_in_free_or_reduced_lunch',
    'percent_households_with_severe_cost_burden',
    'percent_rural',
    'percent_65_and_over',
    'percent_not_proficient_in_english',
    'segregation_index',
    'teen_birth_rate',
    'percent_children_in_single_parent_households',
```

```python
    'percent_low_birthweight',
    'percent_black',
    'rural_urban',
    'food_insecure_lag1',
    'food_insecure_lag2'
]

available_features = [f for f in features if f in df.columns]
df = df[['year', 'fips', 'county.x', 'state.x', 'percent_food_insecure'] + 
 ↪available_features]

print(df.head())

# Step 8: Analyze data

print(f"Years available: {sorted(df['year'].unique())}")
print(f"Counties with data: {df['fips'].nunique()}")

county_years = df.groupby('fips')['year'].count()
print(f"\nMinimum years per county: {county_years.min()}")
print(f"Maximum years per county: {county_years.max()}")

# Step 9: Set n_steps
min_years = county_years.min()
n_steps = min(1, min_years)
print(f"Using n_steps = {n_steps}")

# Step 10: Split data into train and test
latest_year = df['year'].max()
train = df[df['year'] < 2024]
test = df[df['year'] == 2024]

# Step 11: Prepare training data
counties = train['fips'].unique()
X_train, y_train = [], []
scaler = MinMaxScaler()

all_features = train.drop(columns=['year', 'fips', 'county.x', 'state.x',
 ↪'percent_food_insecure'])
scaler.fit(all_features)

for county in counties:
    county_data = train[train['fips'] == county].sort_values('year')
    if len(county_data) < n_steps:
        continue
    features = county_data.drop(columns=['year', 'fips', 'county.x', 'state.x',
 ↪'percent_food_insecure'])
```

```
    target = county_data['percent_food_insecure'].values
    scaled_features = scaler.transform(features)
    for i in range(n_steps, len(county_data)):
        X_train.append(scaled_features[i-n_steps:i])
        y_train.append(target[i])

X_train = np.array(X_train)
y_train = np.array(y_train)

# Step 12: Prepare test data
X_test, y_test = [], []
test_counties = test['fips'].unique()

for county in test_counties:
    county_data = df[(df['fips'] == county) & (df['year'] <= latest_year)].
 ↪sort_values('year')
    if len(county_data) < n_steps + 1:  # Need n_steps years + target year
        continue
    # Get features from n_steps previous years
    features = county_data.iloc[-(n_steps+1):-1].drop(columns=['year', 'fips',␣
 ↪'county.x', 'state.x', 'percent_food_insecure'])
    target = county_data.iloc[-1]['percent_food_insecure']
    scaled_features = scaler.transform(features)
    X_test.append(scaled_features)
    y_test.append(target)

X_test = np.array(X_test)
y_test = np.array(y_test)


# Step 13: Build LSTM model
input_shape = (X_train.shape[1], X_train.shape[2])
model = Sequential([
    LSTM(50, activation='relu', input_shape=input_shape, return_sequences=True),
    Dropout(0.2),
    LSTM(50, activation='relu'),
    Dropout(0.2),
    Dense(1)
])
model.compile(optimizer='adam', loss='mse')

# Step 14: Train the model
history = model.fit(X_train, y_train, epochs=100, batch_size=32,␣
 ↪validation_split=0.2, verbose=1)

# Step 15: Evaluate the model
train_pred = model.predict(X_train)
```

```python
test_pred = model.predict(X_test)
print(f"\nTrain RMSE: {np.sqrt(mean_squared_error(y_train, train_pred))}")
print(f"Test RMSE: {np.sqrt(mean_squared_error(y_test, test_pred))}")

# Step 16: Plot training history
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.legend()
plt.title('Model Training History')
plt.show()
```

```
Step 2: Data loaded
   year   fips   state.x  county.x  \
0  2025  36000  New York     Total
1  2025  36001  New York    Albany
2  2025  36003  New York  Allegany
3  2025  36005  New York     Bronx
4  2025  36007  New York    Broome

   percent_household_income_required_for_child_care_expenses  \
0                                               38.0
1                                               37.0
2                                               43.0
3                                               65.0
4                                               39.0

   food_environment_index  percent_fair_or_poor_health  percent_unemployed  \
0                     8.7                           16                 4.2
1                     8.4                           12                 3.3
2                     8.2                           16                 4.3
3                     7.1                           28                 6.8
4                     7.9                           15                 3.9

   percent_children_in_poverty  percent_severe_housing_problems  …  \
0                           19                               23  …
1                           15                               14  …
2                           17                               12  …
3                           36                               39  …
4                           20                               15  …

   percent_65_and_over  percent_not_proficient_in_english  segregation_index  \
0                 18.6                                  7               0.33
1                 18.7                                  2               0.19
2                 20.9                                  1               0.05
3                 15.3                                 15               0.16
4                 20.7                                  1               0.14

   teen_birth_rate  percent_children_in_single_parent_households  \
```

```
0           10.0                                          26.0
1            8.0                                          27.0
2           10.0                                          19.0
3           17.0                                          52.0
4           12.0                                          25.0

   percent_low_birthweight  percent_black  \
0                      NaN            NaN
1                      NaN            NaN
2                      NaN            NaN
3                      NaN            NaN
4                      NaN            NaN

   percent_children_in_single_parent_households.x  \
0                                             NaN
1                                             NaN
2                                             NaN
3                                             NaN
4                                             NaN

   percent_children_in_single_parent_households.y   rural_urban
0                                             NaN  Mostly Urban
1                                             NaN  Mostly Urban
2                                             NaN  Mostly Rural
3                                             NaN  Mostly Urban
4                                             NaN  Mostly Urban

[5 rows x 33 columns]

Step: Imputed missing values
year                                                        0
fips                                                        0
state.x                                                     0
county.x                                                    0
percent_household_income_required_for_child_care_expenses   0
food_environment_index                                      0
percent_fair_or_poor_health                                 0
percent_unemployed                                          0
percent_children_in_poverty                                 0
percent_severe_housing_problems                             0
percent_completed_high_school                               0
percent_food_insecure                                       0
percent_frequent_mental_distress                            0
percent_uninsured_children                                  0
percent_disconnected_youth                                  0
spending_per_pupil                                          0
school_funding_adequacy                                     0
high_school_graduation_rate                                 0
```

```
median_household_income                                  0
gender_pay_gap                                           0
percent_enrolled_in_free_or_reduced_lunch                0
percent_households_with_severe_cost_burden               0
percent_rural                                            0
percent_65_and_over                                      0
percent_not_proficient_in_english                        0
segregation_index                                        0
teen_birth_rate                                          0
percent_children_in_single_parent_households             0
percent_low_birthweight                                  0
percent_black                                            0
percent_children_in_single_parent_households.x           0
percent_children_in_single_parent_households.y           0
rural_urban                                              0
dtype: int64

Step: Dropped rows with missing 'percent_food_insecure'
   year   fips   state.x  county.x  \
0  2025  36000  New York     Total
1  2025  36001  New York    Albany
2  2025  36003  New York  Allegany
3  2025  36005  New York     Bronx
4  2025  36007  New York    Broome

   percent_household_income_required_for_child_care_expenses  \
0                                               38.0
1                                               37.0
2                                               43.0
3                                               65.0
4                                               39.0

   food_environment_index  percent_fair_or_poor_health  percent_unemployed  \
0                     8.7                           16                 4.2
1                     8.4                           12                 3.3
2                     8.2                           16                 4.3
3                     7.1                           28                 6.8
4                     7.9                           15                 3.9

   percent_children_in_poverty  percent_severe_housing_problems  …  \
0                           19                               23  …
1                           15                               14  …
2                           17                               12  …
3                           36                               39  …
4                           20                               15  …

   percent_65_and_over  percent_not_proficient_in_english  segregation_index  \
0                 18.6                                  7               0.33
```

```
1              18.7                                    2              0.19
2              20.9                                    1              0.05
3              15.3                                   15              0.16
4              20.7                                    1              0.14

   teen_birth_rate  percent_children_in_single_parent_households  \
0             10.0                                          26.0
1              8.0                                          27.0
2             10.0                                          19.0
3             17.0                                          52.0
4             12.0                                          25.0

   percent_low_birthweight  percent_black  \
0                 7.292994       6.134286
1                 7.292994       6.134286
2                 7.292994       6.134286
3                 7.292994       6.134286
4                 7.292994       6.134286

   percent_children_in_single_parent_households.x  \
0                                       22.714286
1                                       22.714286
2                                       22.714286
3                                       22.714286
4                                       22.714286

   percent_children_in_single_parent_households.y   rural_urban
0                                       22.349206  Mostly Urban
1                                       22.349206  Mostly Urban
2                                       22.349206  Mostly Rural
3                                       22.349206  Mostly Urban
4                                       22.349206  Mostly Urban

[5 rows x 33 columns]

Step 3: Dropped rows with missing 'percent_food_insecure'
   year   fips   state.x  county.x  \
0  2025  36000  New York     Total
1  2025  36001  New York    Albany
2  2025  36003  New York  Allegany
3  2025  36005  New York     Bronx
4  2025  36007  New York    Broome

   percent_household_income_required_for_child_care_expenses  \
0                                               38.0
1                                               37.0
2                                               43.0
3                                               65.0
```

```
4                                                39.0

   food_environment_index  percent_fair_or_poor_health  percent_unemployed  \
0                     8.7                           16                 4.2
1                     8.4                           12                 3.3
2                     8.2                           16                 4.3
3                     7.1                           28                 6.8
4                     7.9                           15                 3.9

   percent_children_in_poverty  percent_severe_housing_problems  …  \
0                           19                               23  …
1                           15                               14  …
2                           17                               12  …
3                           36                               39  …
4                           20                               15  …

   percent_65_and_over  percent_not_proficient_in_english  segregation_index  \
0                 18.6                                  7               0.33
1                 18.7                                  2               0.19
2                 20.9                                  1               0.05
3                 15.3                                 15               0.16
4                 20.7                                  1               0.14

   teen_birth_rate  percent_children_in_single_parent_households  \
0             10.0                                          26.0
1              8.0                                          27.0
2             10.0                                          19.0
3             17.0                                          52.0
4             12.0                                          25.0

   percent_low_birthweight  percent_black  \
0                 7.292994       6.134286
1                 7.292994       6.134286
2                 7.292994       6.134286
3                 7.292994       6.134286
4                 7.292994       6.134286

   percent_children_in_single_parent_households.x  \
0                                       22.714286
1                                       22.714286
2                                       22.714286
3                                       22.714286
4                                       22.714286

   percent_children_in_single_parent_households.y   rural_urban
0                                       22.349206  Mostly Urban
1                                       22.349206  Mostly Urban
2                                       22.349206  Mostly Rural
```

```
3                                       22.349206  Mostly Urban
4                                       22.349206  Mostly Urban

[5 rows x 33 columns]

Step 4: Converted 'rural_urban' to numeric
   rural_urban
0            0
1            0
2            1
3            0
4            0

Step 5: Created lag features
      fips  year  percent_food_insecure  food_insecure_lag1  \
315  36000  2020                     11                 NaN
252  36000  2021                     11                11.0
189  36000  2022                     11                11.0
126  36000  2023                     10                11.0
63   36000  2024                     11                10.0

     food_insecure_lag2
315                 NaN
252                 NaN
189                11.0
126                11.0
63                 11.0

Step 6: Dropped rows with missing lag features
     year   fips   state.x county.x  \
189  2022  36000  New York    Total
126  2023  36000  New York    Total
63   2024  36000  New York    Total
0    2025  36000  New York    Total
190  2022  36001  New York   Albany

     percent_household_income_required_for_child_care_expenses  \
189                                           36.26455
126                                           32.00000
63                                            38.00000
0                                             38.00000
190                                           36.26455

     food_environment_index  percent_fair_or_poor_health  percent_unemployed  \
189                     9.0                           16                10.0
126                     8.9                           12                 6.9
63                      8.6                           14                 4.3
0                       8.7                           16                 4.2
```

```
190                     8.3                            15                  7.2

     percent_children_in_poverty  percent_severe_housing_problems  …  \
189                           17                               23  …
126                           19                               23  …
63                            19                               22  …
0                             19                               23  …
190                           13                               15  …

     segregation_index  teen_birth_rate  \
189               0.35             13.0
126               0.34             13.0
63                0.34             11.0
0                 0.33             10.0
190               0.21              9.0

     percent_children_in_single_parent_households  percent_low_birthweight  \
189                                     22.248677                 8.000000
126                                     22.248677                 8.000000
63                                      26.000000                 8.000000
0                                       26.000000                 7.292994
190                                     22.248677                 8.000000

     percent_black  percent_children_in_single_parent_households.x  \
189      14.400000                                       26.000000
126      14.400000                                       26.000000
63       14.400000                                       22.714286
0         6.134286                                       22.714286
190      12.900000                                       29.000000

     percent_children_in_single_parent_households.y  rural_urban  \
189                                       26.000000            0
126                                       26.000000            0
63                                        22.349206            0
0                                         22.349206            0
190                                       27.000000            0

     food_insecure_lag1  food_insecure_lag2
189                11.0                11.0
126                11.0                11.0
63                 10.0                11.0
0                  11.0                10.0
190                10.0                12.0

[5 rows x 35 columns]

Step 7: Selected features
     year   fips county.x   state.x  percent_food_insecure  \
```

```
189  2022  36000    Total  New York                      11
126  2023  36000    Total  New York                      10
63   2024  36000    Total  New York                      11
0    2025  36000    Total  New York                      13
190  2022  36001   Albany  New York                      10

     percent_household_income_required_for_child_care_expenses  \
189                                           36.26455
126                                           32.00000
63                                            38.00000
0                                             38.00000
190                                           36.26455

     food_environment_index  percent_fair_or_poor_health  percent_unemployed  \
189                     9.0                           16                10.0
126                     8.9                           12                 6.9
63                      8.6                           14                 4.3
0                       8.7                           16                 4.2
190                     8.3                           15                 7.2

     percent_children_in_poverty  …  percent_65_and_over  \
189                           17  …                 17.4
126                           19  …                 17.5
63                            19  …                 18.1
0                             19  …                 18.6
190                           13  …                 17.9

     percent_not_proficient_in_english  segregation_index  teen_birth_rate  \
189                                  7               0.35             13.0
126                                  7               0.34             13.0
63                                   7               0.34             11.0
0                                    7               0.33             10.0
190                                  2               0.21              9.0

     percent_children_in_single_parent_households  percent_low_birthweight  \
189                                     22.248677                 8.000000
126                                     22.248677                 8.000000
63                                      26.000000                 8.000000
0                                       26.000000                 7.292994
190                                     22.248677                 8.000000

     percent_black  rural_urban  food_insecure_lag1  food_insecure_lag2
189      14.400000            0                11.0                11.0
126      14.400000            0                11.0                11.0
63       14.400000            0                10.0                11.0
0         6.134286            0                11.0                10.0
190      12.900000            0                10.0                12.0
```

```
[5 rows x 33 columns]

Step 8: Data Analysis
Years available: [2022, 2023, 2024, 2025]
Counties with data: 63

Minimum years per county: 4
Maximum years per county: 4
Using n_steps = 1

Training years: [2022, 2023]
Test year: [2024]

Training data shape: (63, 1, 28)
Training target shape: (63,)

Test data shape: (63, 1, 28)
Test target shape: (63,)
Epoch 1/100
```

```
c:\Users\jashb\Lib\site-packages\keras\src\layers\rnn\rnn.py:200: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When using
Sequential models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(**kwargs)
```

```
2/2             3s 381ms/step - loss:
137.3853 - val_loss: 108.5415
Epoch 2/100
2/2             0s 58ms/step - loss:
139.8515 - val_loss: 108.3155
Epoch 3/100
2/2             0s 59ms/step - loss:
141.4250 - val_loss: 108.1119
Epoch 4/100
2/2             0s 60ms/step - loss:
141.1273 - val_loss: 107.9244
Epoch 5/100
2/2             0s 61ms/step - loss:
140.1974 - val_loss: 107.7381
Epoch 6/100
2/2             0s 75ms/step - loss:
141.7482 - val_loss: 107.5492
Epoch 7/100
2/2             0s 71ms/step - loss:
137.6989 - val_loss: 107.3566
Epoch 8/100
2/2             0s 73ms/step - loss:
132.9959 - val_loss: 107.1525
```

```
Epoch 9/100
2/2            0s 88ms/step - loss:
138.6409 - val_loss: 106.9294
Epoch 10/100
2/2            0s 63ms/step - loss:
136.4078 - val_loss: 106.6821
Epoch 11/100
2/2            0s 69ms/step - loss:
141.0260 - val_loss: 106.4054
Epoch 12/100
2/2            0s 71ms/step - loss:
138.8477 - val_loss: 106.0942
Epoch 13/100
2/2            0s 68ms/step - loss:
135.4921 - val_loss: 105.7401
Epoch 14/100
2/2            0s 68ms/step - loss:
135.8793 - val_loss: 105.3364
Epoch 15/100
2/2            0s 68ms/step - loss:
139.1888 - val_loss: 104.8742
Epoch 16/100
2/2            0s 70ms/step - loss:
139.2800 - val_loss: 104.3432
Epoch 17/100
2/2            0s 67ms/step - loss:
139.3910 - val_loss: 103.7296
Epoch 18/100
2/2            0s 73ms/step - loss:
134.4709 - val_loss: 103.0163
Epoch 19/100
2/2            0s 86ms/step - loss:
135.2823 - val_loss: 102.1841
Epoch 20/100
2/2            0s 76ms/step - loss:
130.6403 - val_loss: 101.2062
Epoch 21/100
2/2            0s 64ms/step - loss:
132.2075 - val_loss: 100.0590
Epoch 22/100
2/2            0s 63ms/step - loss:
130.4311 - val_loss: 98.7142
Epoch 23/100
2/2            0s 66ms/step - loss:
130.3779 - val_loss: 97.1327
Epoch 24/100
2/2            0s 65ms/step - loss:
126.7776 - val_loss: 95.2725
```

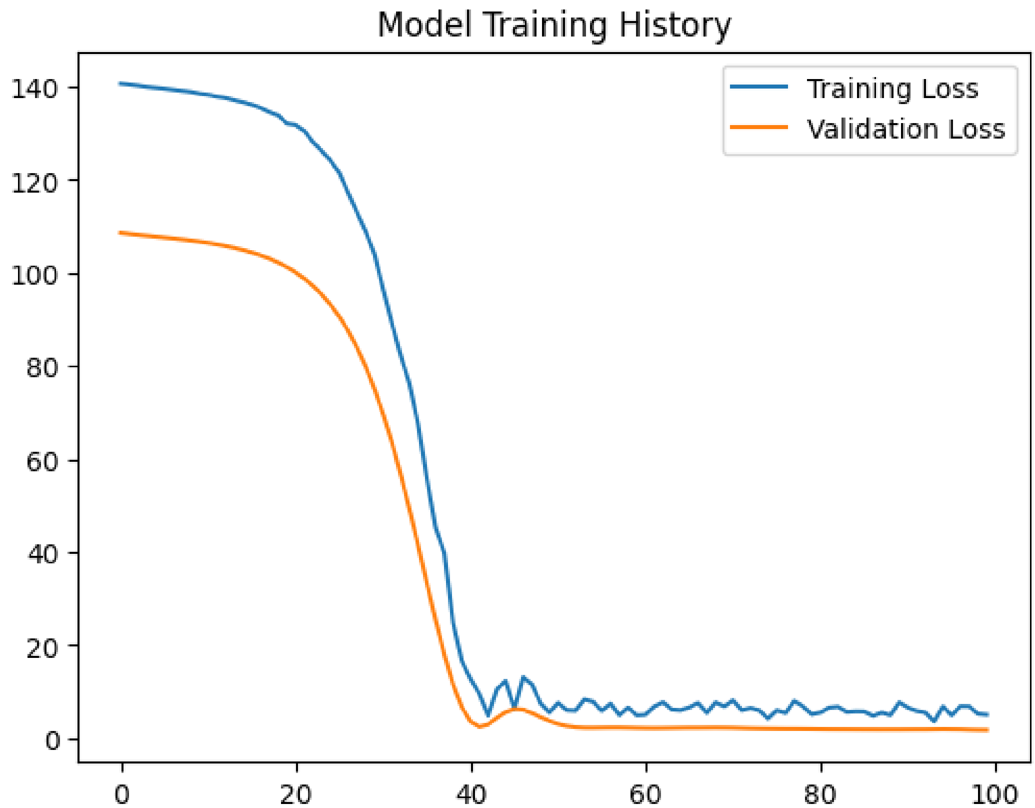
```
Epoch 25/100
2/2            0s 75ms/step - loss:
120.4941 - val_loss: 93.0846
Epoch 26/100
2/2            0s 57ms/step - loss:
123.2378 - val_loss: 90.5079
Epoch 27/100
2/2            0s 57ms/step - loss:
117.8248 - val_loss: 87.4843
Epoch 28/100
2/2            0s 51ms/step - loss:
114.7914 - val_loss: 83.9399
Epoch 29/100
2/2            0s 54ms/step - loss:
109.6894 - val_loss: 79.8177
Epoch 30/100
2/2            0s 52ms/step - loss:
106.7490 - val_loss: 75.0618
Epoch 31/100
2/2            0s 53ms/step - loss:
94.0529 - val_loss: 69.6341
Epoch 32/100
2/2            0s 57ms/step - loss:
89.0301 - val_loss: 63.4924
Epoch 33/100
2/2            0s 54ms/step - loss:
83.1545 - val_loss: 56.6611
Epoch 34/100
2/2            0s 54ms/step - loss:
75.8580 - val_loss: 49.2404
Epoch 35/100
2/2            0s 55ms/step - loss:
69.4846 - val_loss: 41.3763
Epoch 36/100
2/2            0s 50ms/step - loss:
57.1140 - val_loss: 33.2938
Epoch 37/100
2/2            0s 57ms/step - loss:
46.0513 - val_loss: 25.3285
Epoch 38/100
2/2            0s 54ms/step - loss:
40.9223 - val_loss: 17.9013
Epoch 39/100
2/2            0s 56ms/step - loss:
24.7739 - val_loss: 11.5090
Epoch 40/100
2/2            0s 55ms/step - loss:
16.6597 - val_loss: 6.6049
```

```
Epoch 41/100
2/2             0s 53ms/step - loss:
12.6452 - val_loss: 3.5614
Epoch 42/100
2/2             0s 57ms/step - loss:
9.8906 - val_loss: 2.4253
Epoch 43/100
2/2             0s 57ms/step - loss:
4.7565 - val_loss: 2.9436
Epoch 44/100
2/2             0s 54ms/step - loss:
9.8807 - val_loss: 4.3890
Epoch 45/100
2/2             0s 63ms/step - loss:
12.6533 - val_loss: 5.5821
Epoch 46/100
2/2             0s 70ms/step - loss:
5.6062 - val_loss: 6.2701
Epoch 47/100
2/2             0s 66ms/step - loss:
13.3151 - val_loss: 6.1198
Epoch 48/100
2/2             0s 54ms/step - loss:
12.3509 - val_loss: 5.3939
Epoch 49/100
2/2             0s 54ms/step - loss:
8.0004 - val_loss: 4.5215
Epoch 50/100
2/2             0s 54ms/step - loss:
5.7200 - val_loss: 3.7147
Epoch 51/100
2/2             0s 57ms/step - loss:
7.7339 - val_loss: 3.0399
Epoch 52/100
2/2             0s 55ms/step - loss:
5.8826 - val_loss: 2.6077
Epoch 53/100
2/2             0s 61ms/step - loss:
6.4568 - val_loss: 2.3548
Epoch 54/100
2/2             0s 64ms/step - loss:
8.6081 - val_loss: 2.2483
Epoch 55/100
2/2             0s 55ms/step - loss:
7.5514 - val_loss: 2.2409
Epoch 56/100
2/2             0s 57ms/step - loss:
6.0043 - val_loss: 2.2661
```

```
Epoch 57/100
2/2             0s 56ms/step - loss:
7.2024 - val_loss: 2.2870
Epoch 58/100
2/2             0s 62ms/step - loss:
4.9620 - val_loss: 2.2865
Epoch 59/100
2/2             0s 60ms/step - loss:
6.6111 - val_loss: 2.2512
Epoch 60/100
2/2             0s 57ms/step - loss:
4.7760 - val_loss: 2.2050
Epoch 61/100
2/2             0s 59ms/step - loss:
5.2887 - val_loss: 2.1686
Epoch 62/100
2/2             0s 63ms/step - loss:
6.3002 - val_loss: 2.1563
Epoch 63/100
2/2             0s 57ms/step - loss:
7.2110 - val_loss: 2.1702
Epoch 64/100
2/2             0s 54ms/step - loss:
6.0631 - val_loss: 2.1949
Epoch 65/100
2/2             0s 55ms/step - loss:
5.7953 - val_loss: 2.2240
Epoch 66/100
2/2             0s 58ms/step - loss:
6.8793 - val_loss: 2.2406
Epoch 67/100
2/2             0s 56ms/step - loss:
7.4673 - val_loss: 2.2415
Epoch 68/100
2/2             0s 56ms/step - loss:
5.7324 - val_loss: 2.2545
Epoch 69/100
2/2             0s 53ms/step - loss:
8.1572 - val_loss: 2.2658
Epoch 70/100
2/2             0s 59ms/step - loss:
6.9166 - val_loss: 2.2523
Epoch 71/100
2/2             0s 59ms/step - loss:
7.9321 - val_loss: 2.2248
Epoch 72/100
2/2             0s 59ms/step - loss:
5.8552 - val_loss: 2.1643
```

```
Epoch 73/100
2/2            0s 59ms/step - loss:
6.7585 - val_loss: 2.1123
Epoch 74/100
2/2            0s 59ms/step - loss:
5.8720 - val_loss: 2.0809
Epoch 75/100
2/2            0s 55ms/step - loss:
4.6643 - val_loss: 2.0558
Epoch 76/100
2/2            0s 57ms/step - loss:
5.3364 - val_loss: 2.0375
Epoch 77/100
2/2            0s 58ms/step - loss:
5.2693 - val_loss: 2.0170
Epoch 78/100
2/2            0s 56ms/step - loss:
8.1992 - val_loss: 2.0138
Epoch 79/100
2/2            0s 55ms/step - loss:
7.0724 - val_loss: 2.0016
Epoch 80/100
2/2            0s 56ms/step - loss:
4.8750 - val_loss: 1.9831
Epoch 81/100
2/2            0s 62ms/step - loss:
5.4496 - val_loss: 1.9593
Epoch 82/100
2/2            0s 59ms/step - loss:
6.3707 - val_loss: 1.9401
Epoch 83/100
2/2            0s 56ms/step - loss:
6.7137 - val_loss: 1.9302
Epoch 84/100
2/2            0s 57ms/step - loss:
5.2061 - val_loss: 1.9256
Epoch 85/100
2/2            0s 50ms/step - loss:
6.2688 - val_loss: 1.9136
Epoch 86/100
2/2            0s 58ms/step - loss:
5.5633 - val_loss: 1.9033
Epoch 87/100
2/2            0s 55ms/step - loss:
4.5073 - val_loss: 1.8973
Epoch 88/100
2/2            0s 57ms/step - loss:
5.3431 - val_loss: 1.8914
```

```
Epoch 89/100
2/2             0s 55ms/step - loss:
4.9084 - val_loss: 1.8935
Epoch 90/100
2/2             0s 55ms/step - loss:
8.5629 - val_loss: 1.8923
Epoch 91/100
2/2             0s 63ms/step - loss:
6.1700 - val_loss: 1.9011
Epoch 92/100
2/2             0s 57ms/step - loss:
6.1252 - val_loss: 1.9103
Epoch 93/100
2/2             0s 55ms/step - loss:
5.7256 - val_loss: 1.9086
Epoch 94/100
2/2             0s 58ms/step - loss:
3.2019 - val_loss: 1.9362
Epoch 95/100
2/2             0s 59ms/step - loss:
6.0466 - val_loss: 1.9615
Epoch 96/100
2/2             0s 59ms/step - loss:
5.2952 - val_loss: 1.9385
Epoch 97/100
2/2             0s 57ms/step - loss:
6.9181 - val_loss: 1.9166
Epoch 98/100
2/2             0s 59ms/step - loss:
6.6120 - val_loss: 1.8449
Epoch 99/100
2/2             0s 60ms/step - loss:
5.2256 - val_loss: 1.8020
Epoch 100/100
2/2             0s 63ms/step - loss:
5.3380 - val_loss: 1.7803
2/2             0s 193ms/step
2/2             0s 15ms/step

Train RMSE: 1.3436819656524663
Test RMSE: 2.6055313358770573
```



2.1 LSTM 1 : Metrics Table

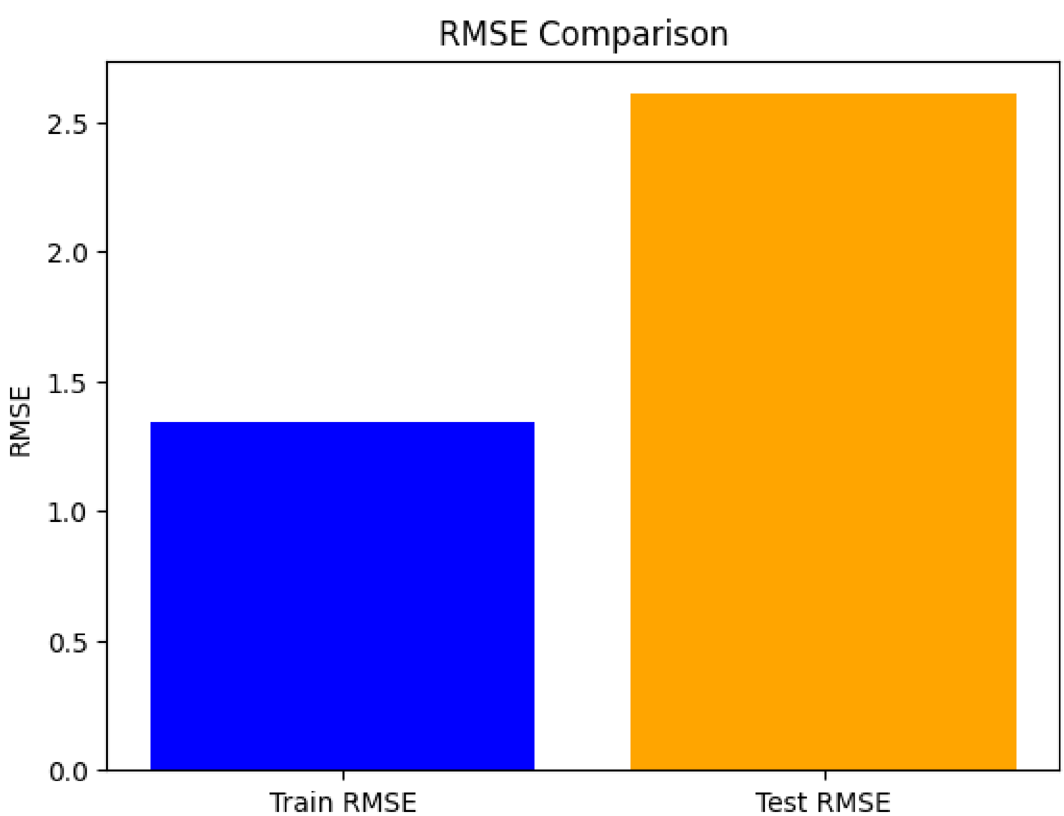
[3]:
```
import pandas as pd

# Calculate MSE
mse_train = mean_squared_error(y_train, train_pred)
mse_test = mean_squared_error(y_test, test_pred)

rmse_train = np.sqrt(mse_train)
rmse_test = np.sqrt(mse_test)

# Calculate MAPE
mape_train = np.mean(np.abs((y_train - train_pred.flatten()) / y_train)) * 100
mape_test = np.mean(np.abs((y_test - test_pred.flatten()) / y_test)) * 100

# Create a table
results = pd.DataFrame({
    "Metric": ["MSE", "RMSE", "MAPE (%)"],
    "Train": [mse_train, rmse_train, mape_train],
    "Test": [mse_test, rmse_test, mape_test]
})
```

```
print("LSTM 4 Model Metrics: Early Stopping\n")
print(results)
```

```
LSTM 4 Model Metrics: Early Stopping

      Metric      Train       Test
0        MSE   1.805481   6.788794
1       RMSE   1.343682   2.605531
2   MAPE (%)  10.120730  17.818789
```

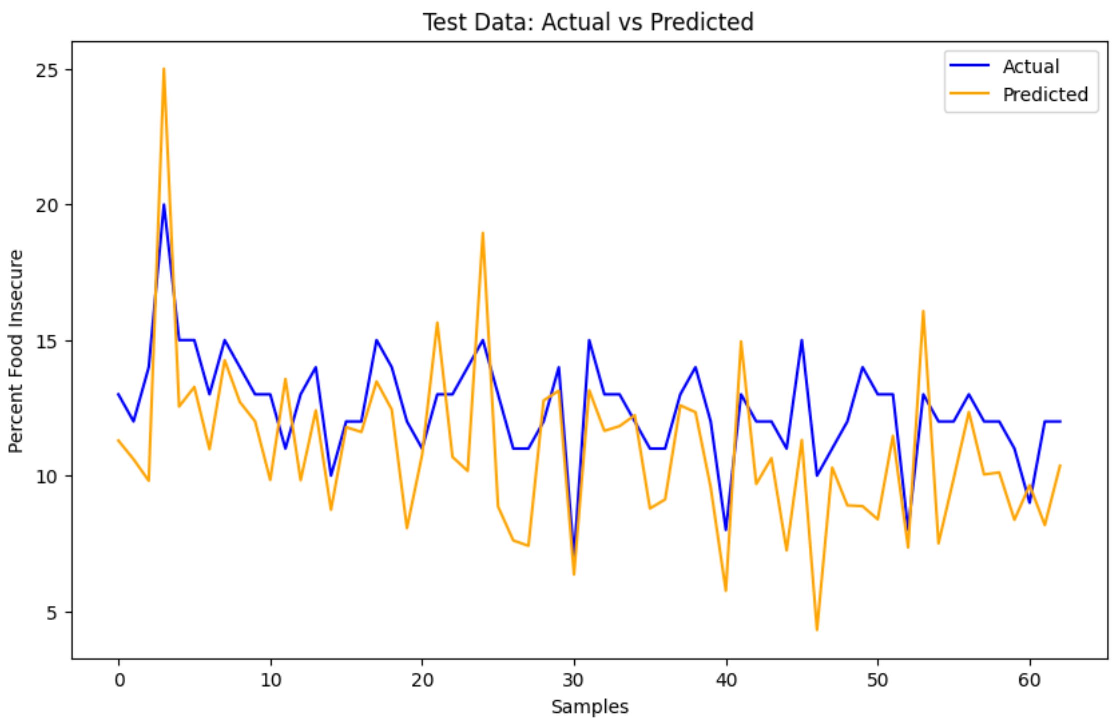
2.2 Visualizing RMSE for LSTM #1

[4]:
```
# Visualize RMSE
train_rmse = np.sqrt(mean_squared_error(y_train, train_pred))
test_rmse = np.sqrt(mean_squared_error(y_test, test_pred))

plt.bar(['Train RMSE', 'Test RMSE'], [train_rmse, test_rmse], color=['blue',␣
 ↪'orange'])
plt.title('RMSE Comparison')
plt.ylabel('RMSE')
plt.show()
```



2.3 Test Data : Predicted vs Actual

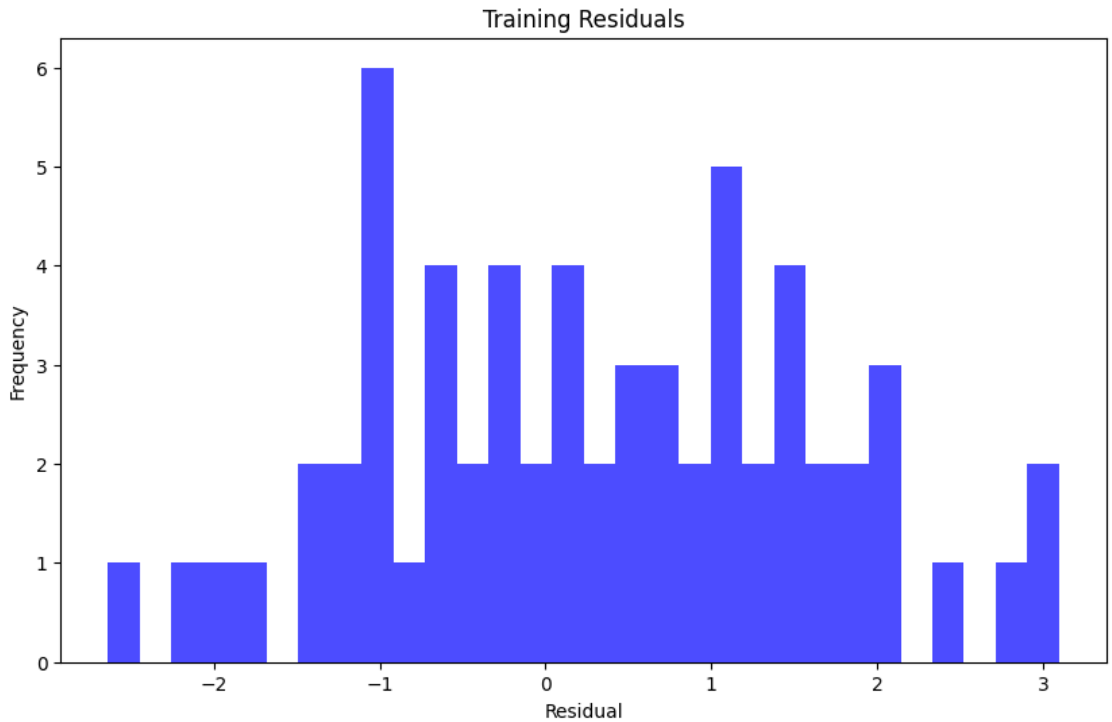
[5]:
```
plt.figure(figsize=(10, 6))
plt.plot(y_test, label='Actual', color='blue')
plt.plot(test_pred, label='Predicted', color='orange')
plt.title('Test Data: Actual vs Predicted')
plt.xlabel('Samples')
plt.ylabel('Percent Food Insecure')
plt.legend()
plt.show()
```



2.4 Residual Analysis

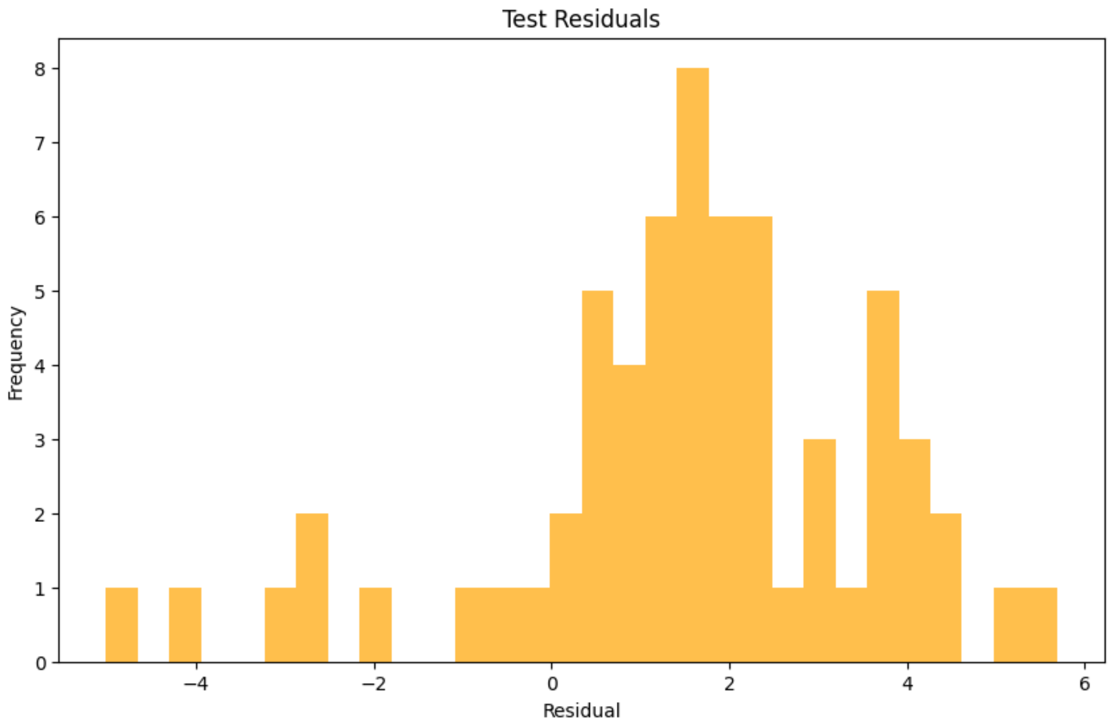
2.4.1 Training Residuals

[6]:
```
train_residuals = y_train - train_pred.flatten()
plt.figure(figsize=(10, 6))
plt.hist(train_residuals, bins=30, color='blue', alpha=0.7)
plt.title('Training Residuals')
plt.xlabel('Residual')
plt.ylabel('Frequency')
plt.show()
```



2.4.2 Test Residuals

[7]:
```
test_residuals = y_test - test_pred.flatten()
plt.figure(figsize=(10, 6))
plt.hist(test_residuals, bins=30, color='orange', alpha=0.7)
plt.title('Test Residuals')
plt.xlabel('Residual')
plt.ylabel('Frequency')
plt.show()
```

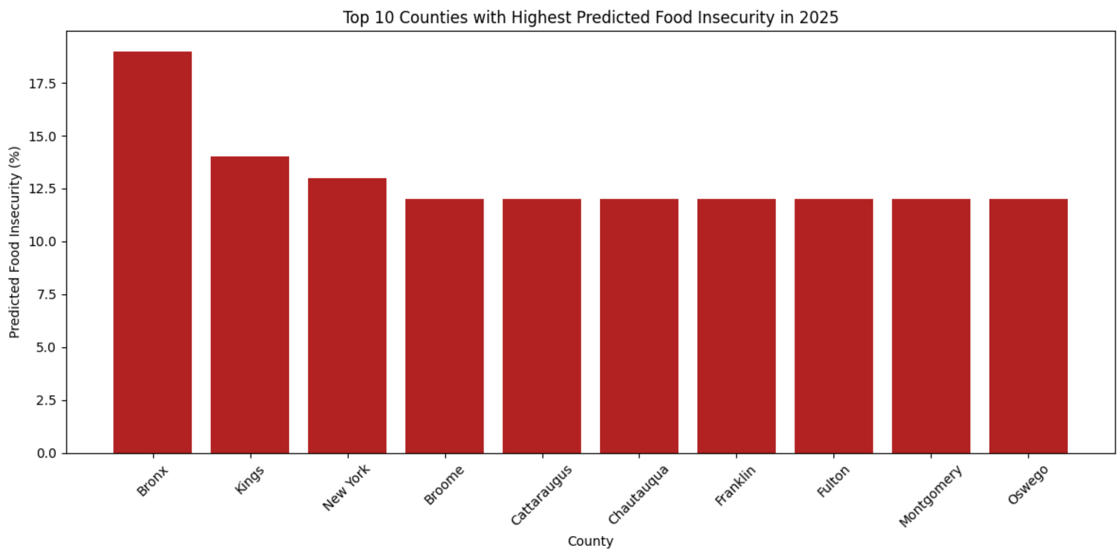


2.5 Training Loss Curve

[8]:
```
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.legend()
plt.title('Model Training History')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.show()
```

2.6 Highest percentage yearly change in food insecurity

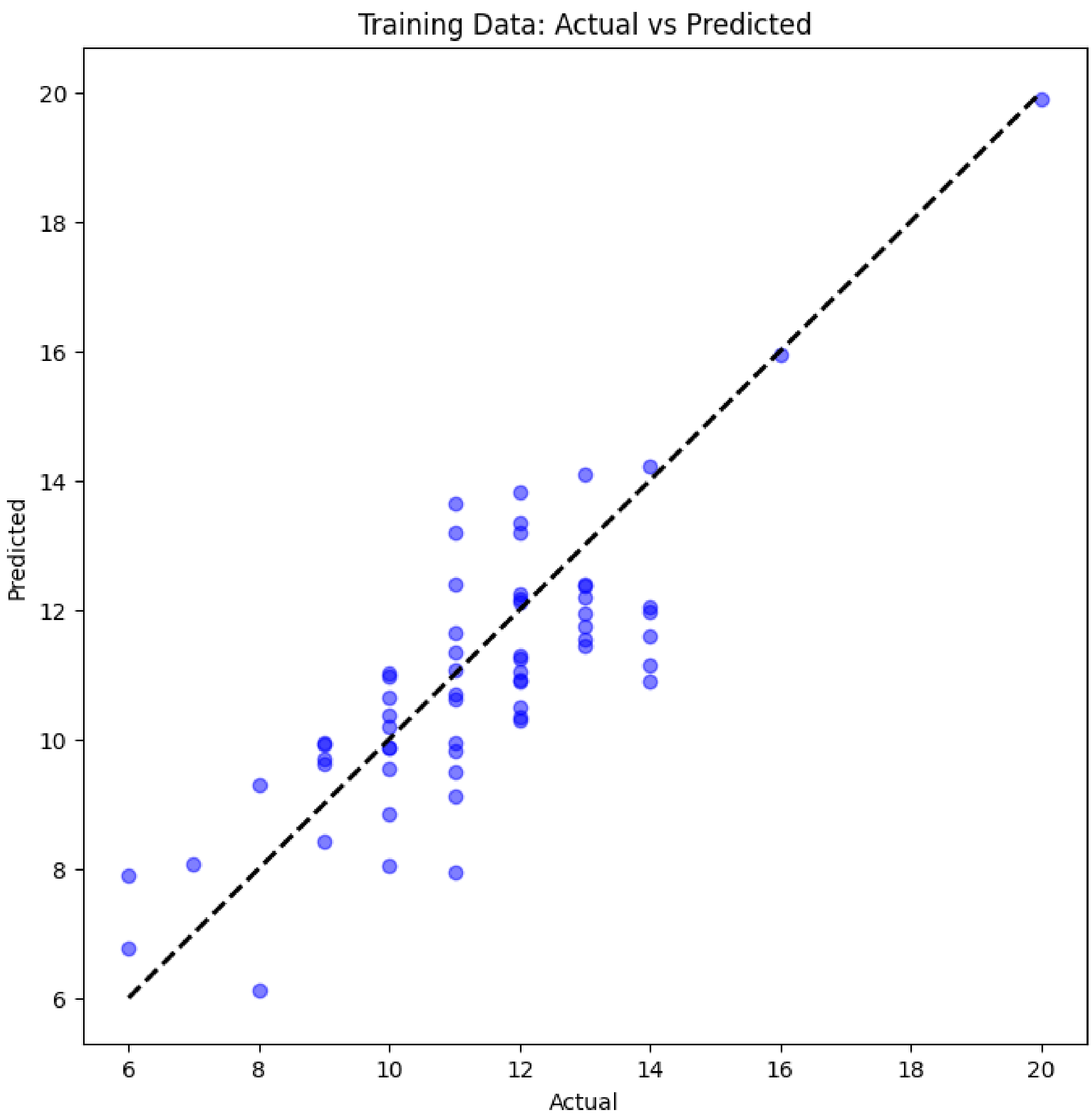
[9]:
```
top_10_counties = test.nlargest(10, 'percent_food_insecure')

plt.figure(figsize=(12, 6))
plt.bar(top_10_counties['county.x'], top_10_counties['percent_food_insecure'],␣
 ↪color='firebrick')
plt.title('Top 10 Counties with Highest Predicted Food Insecurity in 2025')
plt.xlabel('County')
plt.ylabel('Predicted Food Insecurity (%)')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



2.7 Scatter Plot Analysis: Actual vs Predicted

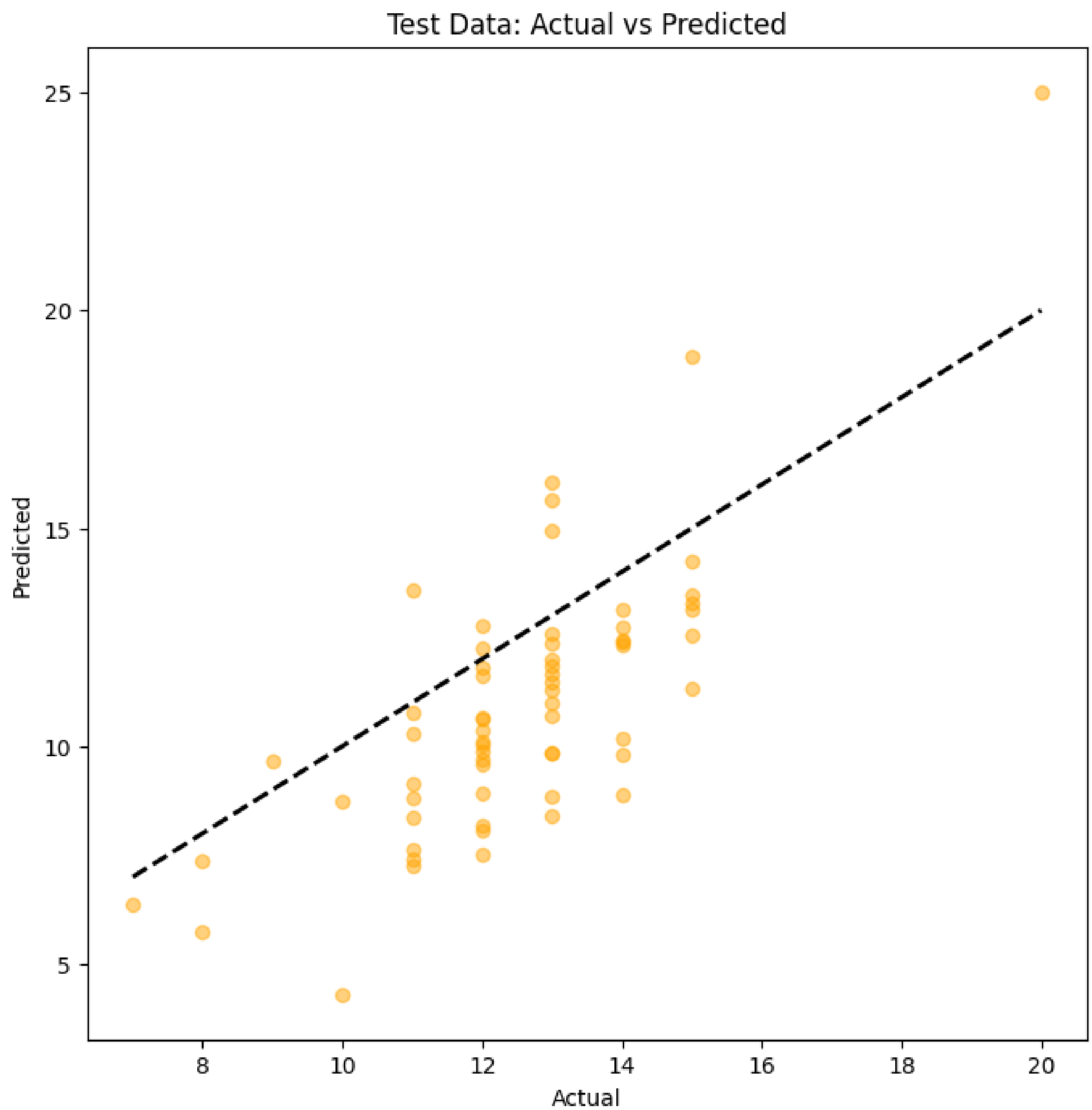
2.7.1 Training Scatter

[10]:
```
plt.figure(figsize=(8, 8))
plt.scatter(y_train, train_pred, alpha=0.5, color='blue')
plt.plot([y_train.min(), y_train.max()], [y_train.min(), y_train.max()], 'k--',␣
 ↪lw=2)
plt.title('Training Data: Actual vs Predicted')
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.show()
```



2.7.2 Test Scatter

[11]:
```
plt.figure(figsize=(8, 8))
plt.scatter(y_test, test_pred, alpha=0.5, color='orange')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--',␣
    ↪lw=2)
plt.title('Test Data: Actual vs Predicted')
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.show()
```



3 LSTM Experiment 2:

- This LSTM model attempts to increase the amount of n_steps which will bring in more historical data into the prediction. Aiding the model in leaning temporal patterns more effectively

[ ]:
```
# Step 2: Load data
filepath = "C:\\Users\\jashb\\OneDrive\\Documents\\Masters Data Science\\Spring␣
↪2025\\DATA 698\\Masters Project\\final_data.csv"
df = pd.read_csv(filepath)

print(df.head())
```

```python
numeric_columns = df.select_dtypes(include=['float64', 'int64']).columns
df[numeric_columns] = df[numeric_columns].fillna(df[numeric_columns].mean())

# Impute categorical columns with the most frequent value
categorical_columns = df.select_dtypes(include=['object']).columns
df[categorical_columns] = df[categorical_columns].
↪fillna(df[categorical_columns].mode().iloc[0])


print(df.isnull().sum())  # Verify no missing values remain

# Step: Drop rows with missing target
df = df.dropna(subset=['percent_food_insecure'])

print(df.head())
# Step 3: Drop rows with missing target
df = df.dropna(subset=['percent_food_insecure'])

print(df.head())

# Step 4: Convert 'rural_urban' to numeric
df['rural_urban'] = pd.factorize(df['rural_urban'])[0]

print(df[['rural_urban']].head())

# Step 5: Create lag features
df = df.sort_values(['fips', 'year'])
df['food_insecure_lag1'] = df.groupby('fips')['percent_food_insecure'].shift(1)
df['food_insecure_lag2'] = df.groupby('fips')['percent_food_insecure'].shift(2)

print(df[['fips', 'year', 'percent_food_insecure', 'food_insecure_lag1',␣
↪'food_insecure_lag2']].head())

# Step 6: Drop rows with missing lag features
df = df.dropna(subset=['food_insecure_lag1', 'food_insecure_lag2'])

print(df.head())

# Step 7: Select features
features = [
    'percent_household_income_required_for_child_care_expenses',
    'food_environment_index',
    'percent_fair_or_poor_health',
    'percent_unemployed',
    'percent_children_in_poverty',
    'percent_severe_housing_problems',
```

```
    'percent_completed_high_school',
    'percent_frequent_mental_distress',
    'percent_uninsured_children',
    'percent_disconnected_youth',
    'spending_per_pupil',
    'school_funding_adequacy',
    'high_school_graduation_rate',
    'median_household_income',
    'gender_pay_gap',
    'percent_enrolled_in_free_or_reduced_lunch',
    'percent_households_with_severe_cost_burden',
    'percent_rural',
    'percent_65_and_over',
    'percent_not_proficient_in_english',
    'segregation_index',
    'teen_birth_rate',
    'percent_children_in_single_parent_households',
    'percent_low_birthweight',
    'percent_black',
    'rural_urban',
    'food_insecure_lag1',
    'food_insecure_lag2'
]

available_features = [f for f in features if f in df.columns]
df = df[['year', 'fips', 'county.x', 'state.x', 'percent_food_insecure'] +
 ↪available_features]

print(df.head())

# Step 8: Analyze data

county_years = df.groupby('fips')['year'].count()


# Step 9: Set n_steps
min_years = county_years.min()
n_steps = min(1, min_years)  # Use 3 if possible, otherwise use the minimum
 ↪available

# Step 10: Split data into train and test
latest_year = df['year'].max()
train = df[df['year'] < 2024]
test = df[df['year'] == 2024]

# Step 11: Prepare training data
counties = train['fips'].unique()
```

```
X_train, y_train = [], []
scaler = MinMaxScaler()

all_features = train.drop(columns=['year', 'fips', 'county.x', 'state.x',␣
↪'percent_food_insecure'])
scaler.fit(all_features)

for county in counties:
    county_data = train[train['fips'] == county].sort_values('year')
    if len(county_data) < n_steps:
        continue
    features = county_data.drop(columns=['year', 'fips', 'county.x', 'state.x',␣
↪'percent_food_insecure'])
    target = county_data['percent_food_insecure'].values
    scaled_features = scaler.transform(features)
    for i in range(n_steps, len(county_data)):
        X_train.append(scaled_features[i-n_steps:i])
        y_train.append(target[i])

X_train = np.array(X_train)
y_train = np.array(y_train)

# Step 12: Prepare test data
X_test, y_test = [], []
test_counties = test['fips'].unique()

for county in test_counties:
    county_data = df[(df['fips'] == county) & (df['year'] <= latest_year)].
↪sort_values('year')
    if len(county_data) < n_steps + 1:  # Need n_steps years + target year
        continue
    # Get features from n_steps previous years
    features = county_data.iloc[-(n_steps+1):-1].drop(columns=['year', 'fips',␣
↪'county.x', 'state.x', 'percent_food_insecure'])
    target = county_data.iloc[-1]['percent_food_insecure']
    scaled_features = scaler.transform(features)
    X_test.append(scaled_features)
    y_test.append(target)

X_test = np.array(X_test)
y_test = np.array(y_test)


# Step 13: Build LSTM model
input_shape = (X_train.shape[1], X_train.shape[2])
model = Sequential([
    LSTM(50, activation='relu', input_shape=input_shape, return_sequences=True),
```

```
    Dropout(0.2),
    LSTM(50, activation='relu'),
    Dropout(0.2),
    Dense(1)
])
model.compile(optimizer='adam', loss='mse')

# Step 14: Train the model
history = model.fit(X_train, y_train, epochs=100, batch_size=32,␣
↪validation_split=0.2, verbose=1)

# Step 15: Evaluate the model
train_pred = model.predict(X_train)
test_pred = model.predict(X_test)
print(f"\nTrain RMSE: {np.sqrt(mean_squared_error(y_train, train_pred))}")
print(f"Test RMSE: {np.sqrt(mean_squared_error(y_test, test_pred))}")

# Step 16: Plot training history
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.legend()
plt.title('Model Training History')
plt.show()
```

```
Step 2: Data loaded
   year   fips   state.x   county.x  \
0  2025  36000  New York      Total
1  2025  36001  New York     Albany
2  2025  36003  New York   Allegany
3  2025  36005  New York      Bronx
4  2025  36007  New York     Broome

   percent_household_income_required_for_child_care_expenses  \
0                                               38.0
1                                               37.0
2                                               43.0
3                                               65.0
4                                               39.0

   food_environment_index  percent_fair_or_poor_health  percent_unemployed  \
0                     8.7                           16                 4.2
1                     8.4                           12                 3.3
2                     8.2                           16                 4.3
3                     7.1                           28                 6.8
4                     7.9                           15                 3.9

   percent_children_in_poverty  percent_severe_housing_problems  …  \
0                           19                               23  …
```

```
1                           15                                        14  …
2                           17                                        12  …
3                           36                                        39  …
4                           20                                        15  …

   percent_65_and_over  percent_not_proficient_in_english  segregation_index  \
0                 18.6                                  7               0.33
1                 18.7                                  2               0.19
2                 20.9                                  1               0.05
3                 15.3                                 15               0.16
4                 20.7                                  1               0.14

   teen_birth_rate  percent_children_in_single_parent_households  \
0             10.0                                          26.0
1              8.0                                          27.0
2             10.0                                          19.0
3             17.0                                          52.0
4             12.0                                          25.0

   percent_low_birthweight  percent_black  \
0                      NaN            NaN
1                      NaN            NaN
2                      NaN            NaN
3                      NaN            NaN
4                      NaN            NaN

   percent_children_in_single_parent_households.x  \
0                                             NaN
1                                             NaN
2                                             NaN
3                                             NaN
4                                             NaN

   percent_children_in_single_parent_households.y   rural_urban
0                                             NaN  Mostly Urban
1                                             NaN  Mostly Urban
2                                             NaN  Mostly Rural
3                                             NaN  Mostly Urban
4                                             NaN  Mostly Urban

[5 rows x 33 columns]

Step: Imputed missing values
year                                                            0
fips                                                            0
state.x                                                         0
county.x                                                        0
percent_household_income_required_for_child_care_expenses       0
```

```
food_environment_index                                      0
percent_fair_or_poor_health                                 0
percent_unemployed                                          0
percent_children_in_poverty                                 0
percent_severe_housing_problems                             0
percent_completed_high_school                               0
percent_food_insecure                                       0
percent_frequent_mental_distress                            0
percent_uninsured_children                                  0
percent_disconnected_youth                                  0
spending_per_pupil                                          0
school_funding_adequacy                                     0
high_school_graduation_rate                                 0
median_household_income                                     0
gender_pay_gap                                              0
percent_enrolled_in_free_or_reduced_lunch                   0
percent_households_with_severe_cost_burden                  0
percent_rural                                               0
percent_65_and_over                                         0
percent_not_proficient_in_english                           0
segregation_index                                           0
teen_birth_rate                                             0
percent_children_in_single_parent_households                0
percent_low_birthweight                                     0
percent_black                                               0
percent_children_in_single_parent_households.x              0
percent_children_in_single_parent_households.y              0
rural_urban                                                 0
dtype: int64

Step: Dropped rows with missing 'percent_food_insecure'
   year   fips   state.x  county.x  \
0  2025  36000  New York     Total
1  2025  36001  New York    Albany
2  2025  36003  New York  Allegany
3  2025  36005  New York     Bronx
4  2025  36007  New York    Broome

   percent_household_income_required_for_child_care_expenses  \
0                                               38.0
1                                               37.0
2                                               43.0
3                                               65.0
4                                               39.0

   food_environment_index  percent_fair_or_poor_health  percent_unemployed  \
0                     8.7                           16                 4.2
1                     8.4                           12                 3.3
```

```
2                    8.2                            16                 4.3
3                    7.1                            28                 6.8
4                    7.9                            15                 3.9

   percent_children_in_poverty  percent_severe_housing_problems  …  \
0                           19                               23  …
1                           15                               14  …
2                           17                               12  …
3                           36                               39  …
4                           20                               15  …

   percent_65_and_over  percent_not_proficient_in_english  segregation_index  \
0                 18.6                                  7               0.33
1                 18.7                                  2               0.19
2                 20.9                                  1               0.05
3                 15.3                                 15               0.16
4                 20.7                                  1               0.14

   teen_birth_rate  percent_children_in_single_parent_households  \
0             10.0                                          26.0
1              8.0                                          27.0
2             10.0                                          19.0
3             17.0                                          52.0
4             12.0                                          25.0

   percent_low_birthweight  percent_black  \
0                 7.292994       6.134286
1                 7.292994       6.134286
2                 7.292994       6.134286
3                 7.292994       6.134286
4                 7.292994       6.134286

   percent_children_in_single_parent_households.x  \
0                                       22.714286
1                                       22.714286
2                                       22.714286
3                                       22.714286
4                                       22.714286

   percent_children_in_single_parent_households.y   rural_urban
0                                       22.349206  Mostly Urban
1                                       22.349206  Mostly Urban
2                                       22.349206  Mostly Rural
3                                       22.349206  Mostly Urban
4                                       22.349206  Mostly Urban

[5 rows x 33 columns]
```

```
Step 3: Dropped rows with missing 'percent_food_insecure'
   year   fips   state.x  county.x  \
0  2025  36000  New York     Total
1  2025  36001  New York    Albany
2  2025  36003  New York  Allegany
3  2025  36005  New York     Bronx
4  2025  36007  New York    Broome

   percent_household_income_required_for_child_care_expenses  \
0                                               38.0
1                                               37.0
2                                               43.0
3                                               65.0
4                                               39.0

   food_environment_index  percent_fair_or_poor_health  percent_unemployed  \
0                     8.7                           16                 4.2
1                     8.4                           12                 3.3
2                     8.2                           16                 4.3
3                     7.1                           28                 6.8
4                     7.9                           15                 3.9

   percent_children_in_poverty  percent_severe_housing_problems  …  \
0                           19                               23  …
1                           15                               14  …
2                           17                               12  …
3                           36                               39  …
4                           20                               15  …

   percent_65_and_over  percent_not_proficient_in_english  segregation_index  \
0                 18.6                                  7               0.33
1                 18.7                                  2               0.19
2                 20.9                                  1               0.05
3                 15.3                                 15               0.16
4                 20.7                                  1               0.14

   teen_birth_rate  percent_children_in_single_parent_households  \
0             10.0                                          26.0
1              8.0                                          27.0
2             10.0                                          19.0
3             17.0                                          52.0
4             12.0                                          25.0

   percent_low_birthweight  percent_black  \
0                 7.292994       6.134286
1                 7.292994       6.134286
2                 7.292994       6.134286
3                 7.292994       6.134286
```

```
4                 7.292994       6.134286

   percent_children_in_single_parent_households.x  \
0                                       22.714286
1                                       22.714286
2                                       22.714286
3                                       22.714286
4                                       22.714286

   percent_children_in_single_parent_households.y   rural_urban
0                                       22.349206  Mostly Urban
1                                       22.349206  Mostly Urban
2                                       22.349206  Mostly Rural
3                                       22.349206  Mostly Urban
4                                       22.349206  Mostly Urban

[5 rows x 33 columns]

Step 4: Converted 'rural_urban' to numeric
   rural_urban
0            0
1            0
2            1
3            0
4            0

Step 5: Created lag features
      fips  year  percent_food_insecure  food_insecure_lag1  \
315  36000  2020                     11                 NaN
252  36000  2021                     11                11.0
189  36000  2022                     11                11.0
126  36000  2023                     10                11.0
63   36000  2024                     11                10.0

     food_insecure_lag2
315                 NaN
252                 NaN
189                11.0
126                11.0
63                 11.0

Step 6: Dropped rows with missing lag features
     year   fips   state.x county.x  \
189  2022  36000  New York    Total
126  2023  36000  New York    Total
63   2024  36000  New York    Total
0    2025  36000  New York    Total
190  2022  36001  New York   Albany
```

```
     percent_household_income_required_for_child_care_expenses  \
189                                           36.26455
126                                           32.00000
63                                            38.00000
0                                             38.00000
190                                           36.26455

     food_environment_index  percent_fair_or_poor_health  percent_unemployed  \
189                     9.0                           16                10.0
126                     8.9                           12                 6.9
63                      8.6                           14                 4.3
0                       8.7                           16                 4.2
190                     8.3                           15                 7.2

     percent_children_in_poverty  percent_severe_housing_problems  …  \
189                           17                               23  …
126                           19                               23  …
63                            19                               22  …
0                             19                               23  …
190                           13                               15  …

     segregation_index  teen_birth_rate  \
189               0.35             13.0
126               0.34             13.0
63                0.34             11.0
0                 0.33             10.0
190               0.21              9.0

     percent_children_in_single_parent_households  percent_low_birthweight  \
189                                     22.248677                 8.000000
126                                     22.248677                 8.000000
63                                      26.000000                 8.000000
0                                       26.000000                 7.292994
190                                     22.248677                 8.000000

     percent_black  percent_children_in_single_parent_households.x  \
189      14.400000                                       26.000000
126      14.400000                                       26.000000
63       14.400000                                       22.714286
0         6.134286                                       22.714286
190      12.900000                                       29.000000

     percent_children_in_single_parent_households.y  rural_urban  \
189                                       26.000000            0
126                                       26.000000            0
63                                        22.349206            0
0                                         22.349206            0
```

```
190                                      27.000000            0

     food_insecure_lag1  food_insecure_lag2
189                11.0                11.0
126                11.0                11.0
63                 10.0                11.0
0                  11.0                10.0
190                10.0                12.0

[5 rows x 35 columns]

Step 7: Selected features
     year   fips county.x   state.x  percent_food_insecure  \
189  2022  36000    Total  New York                     11
126  2023  36000    Total  New York                     10
63   2024  36000    Total  New York                     11
0    2025  36000    Total  New York                     13
190  2022  36001   Albany  New York                     10

     percent_household_income_required_for_child_care_expenses  \
189                                           36.26455
126                                           32.00000
63                                            38.00000
0                                             38.00000
190                                           36.26455

     food_environment_index  percent_fair_or_poor_health  percent_unemployed  \
189                     9.0                           16                10.0
126                     8.9                           12                 6.9
63                      8.6                           14                 4.3
0                       8.7                           16                 4.2
190                     8.3                           15                 7.2

     percent_children_in_poverty  …  percent_65_and_over  \
189                           17  …                 17.4
126                           19  …                 17.5
63                            19  …                 18.1
0                             19  …                 18.6
190                           13  …                 17.9

     percent_not_proficient_in_english  segregation_index  teen_birth_rate  \
189                                  7               0.35             13.0
126                                  7               0.34             13.0
63                                   7               0.34             11.0
0                                    7               0.33             10.0
190                                  2               0.21              9.0

     percent_children_in_single_parent_households  percent_low_birthweight  \
```

```
189                                         22.248677                  8.000000
126                                         22.248677                  8.000000
63                                          26.000000                  8.000000
0                                           26.000000                  7.292994
190                                         22.248677                  8.000000

     percent_black  rural_urban  food_insecure_lag1  food_insecure_lag2
189      14.400000            0                11.0                11.0
126      14.400000            0                11.0                11.0
63       14.400000            0                10.0                11.0
0         6.134286            0                11.0                10.0
190      12.900000            0                10.0                12.0

[5 rows x 33 columns]

Step 8: Data Analysis
Years available: [2022, 2023, 2024, 2025]
Counties with data: 63

Minimum years per county: 4
Maximum years per county: 4
Using n_steps = 1

Training years: [2022, 2023]
Test year: [2024]

Training data shape: (63, 1, 28)
Training target shape: (63,)

Test data shape: (63, 1, 28)
Test target shape: (63,)
Epoch 1/100
```

```
c:\Users\jashb\Lib\site-packages\keras\src\layers\rnn\rnn.py:200: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When using
Sequential models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(**kwargs)
```

```
2/2               2s 309ms/step - loss:
141.2389 - val_loss: 108.1757
Epoch 2/100
2/2               0s 51ms/step - loss:
142.5439 - val_loss: 107.9701
Epoch 3/100
2/2               0s 60ms/step - loss:
139.8260 - val_loss: 107.7509
Epoch 4/100
2/2               0s 58ms/step - loss:
```

```
137.5416 - val_loss: 107.5169
Epoch 5/100
2/2             0s 62ms/step - loss:
142.2080 - val_loss: 107.2652
Epoch 6/100
2/2             0s 63ms/step - loss:
137.3421 - val_loss: 106.9858
Epoch 7/100
2/2             0s 64ms/step - loss:
134.0715 - val_loss: 106.6769
Epoch 8/100
2/2             0s 63ms/step - loss:
139.3503 - val_loss: 106.3258
Epoch 9/100
2/2             0s 62ms/step - loss:
139.6154 - val_loss: 105.9310
Epoch 10/100
2/2             0s 66ms/step - loss:
136.4764 - val_loss: 105.4865
Epoch 11/100
2/2             0s 65ms/step - loss:
135.2802 - val_loss: 104.9833
Epoch 12/100
2/2             0s 67ms/step - loss:
136.5366 - val_loss: 104.4087
Epoch 13/100
2/2             0s 62ms/step - loss:
134.5749 - val_loss: 103.7477
Epoch 14/100
2/2             0s 54ms/step - loss:
136.6111 - val_loss: 102.9814
Epoch 15/100
2/2             0s 57ms/step - loss:
132.4516 - val_loss: 102.0863
Epoch 16/100
2/2             0s 54ms/step - loss:
130.0177 - val_loss: 101.0494
Epoch 17/100
2/2             0s 51ms/step - loss:
132.7941 - val_loss: 99.8523
Epoch 18/100
2/2             0s 52ms/step - loss:
131.3441 - val_loss: 98.4583
Epoch 19/100
2/2             0s 52ms/step - loss:
128.5053 - val_loss: 96.8328
Epoch 20/100
2/2             0s 53ms/step - loss:
```

```
126.4136 - val_loss: 94.9319
Epoch 21/100
2/2              0s 55ms/step - loss:
121.8986 - val_loss: 92.7079
Epoch 22/100
2/2              0s 61ms/step - loss:
120.1742 - val_loss: 90.1039
Epoch 23/100
2/2              0s 55ms/step - loss:
119.6669 - val_loss: 87.0607
Epoch 24/100
2/2              0s 53ms/step - loss:
114.7009 - val_loss: 83.5055
Epoch 25/100
2/2              0s 54ms/step - loss:
106.5179 - val_loss: 79.3659
Epoch 26/100
2/2              0s 51ms/step - loss:
105.6465 - val_loss: 74.5763
Epoch 27/100
2/2              0s 53ms/step - loss:
99.2586 - val_loss: 69.0884
Epoch 28/100
2/2              0s 49ms/step - loss:
89.5585 - val_loss: 62.8389
Epoch 29/100
2/2              0s 54ms/step - loss:
82.3382 - val_loss: 55.8049
Epoch 30/100
2/2              0s 53ms/step - loss:
70.9836 - val_loss: 48.0262
Epoch 31/100
2/2              0s 55ms/step - loss:
63.4070 - val_loss: 39.6372
Epoch 32/100
2/2              0s 63ms/step - loss:
51.4981 - val_loss: 30.9247
Epoch 33/100
2/2              0s 64ms/step - loss:
45.8382 - val_loss: 22.3248
Epoch 34/100
2/2              0s 50ms/step - loss:
32.8848 - val_loss: 14.4852
Epoch 35/100
2/2              0s 70ms/step - loss:
21.7567 - val_loss: 8.1456
Epoch 36/100
2/2              0s 57ms/step - loss:
```

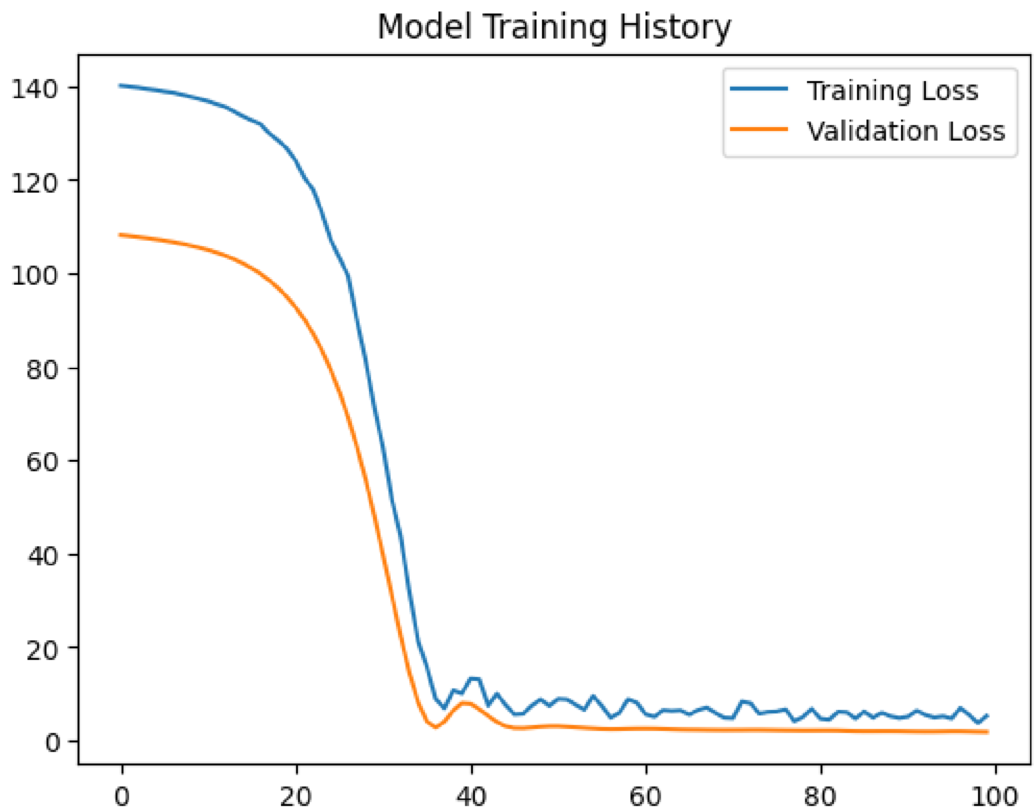
```
15.6746 - val_loss: 4.0576
Epoch 37/100
2/2              0s 57ms/step - loss:
9.5450 - val_loss: 2.7686
Epoch 38/100
2/2              0s 57ms/step - loss:
7.0861 - val_loss: 4.0104
Epoch 39/100
2/2              0s 60ms/step - loss:
9.5697 - val_loss: 6.3814
Epoch 40/100
2/2              0s 57ms/step - loss:
9.7935 - val_loss: 7.9835
Epoch 41/100
2/2              0s 52ms/step - loss:
14.7711 - val_loss: 7.8449
Epoch 42/100
2/2              0s 55ms/step - loss:
13.1237 - val_loss: 6.6635
Epoch 43/100
2/2              0s 59ms/step - loss:
7.3820 - val_loss: 5.2474
Epoch 44/100
2/2              0s 61ms/step - loss:
10.2956 - val_loss: 3.9515
Epoch 45/100
2/2              0s 62ms/step - loss:
7.5783 - val_loss: 3.0513
Epoch 46/100
2/2              0s 56ms/step - loss:
5.0819 - val_loss: 2.6586
Epoch 47/100
2/2              0s 58ms/step - loss:
5.9289 - val_loss: 2.6262
Epoch 48/100
2/2              0s 57ms/step - loss:
7.6009 - val_loss: 2.7766
Epoch 49/100
2/2              0s 57ms/step - loss:
9.2343 - val_loss: 2.9438
Epoch 50/100
2/2              0s 58ms/step - loss:
6.5246 - val_loss: 3.0542
Epoch 51/100
2/2              0s 54ms/step - loss:
8.5480 - val_loss: 3.0588
Epoch 52/100
2/2              0s 55ms/step - loss:
```

```
8.8140 - val_loss: 2.9367
Epoch 53/100
2/2              0s 55ms/step - loss:
7.7444 - val_loss: 2.8147
Epoch 54/100
2/2              0s 56ms/step - loss:
7.0289 - val_loss: 2.6839
Epoch 55/100
2/2              0s 52ms/step - loss:
9.9974 - val_loss: 2.5630
Epoch 56/100
2/2              0s 59ms/step - loss:
7.7945 - val_loss: 2.4585
Epoch 57/100
2/2              0s 61ms/step - loss:
4.7489 - val_loss: 2.4120
Epoch 58/100
2/2              0s 54ms/step - loss:
5.7001 - val_loss: 2.4334
Epoch 59/100
2/2              0s 50ms/step - loss:
9.4205 - val_loss: 2.4879
Epoch 60/100
2/2              0s 57ms/step - loss:
8.9270 - val_loss: 2.5216
Epoch 61/100
2/2              0s 50ms/step - loss:
5.5395 - val_loss: 2.5293
Epoch 62/100
2/2              0s 54ms/step - loss:
4.6395 - val_loss: 2.5119
Epoch 63/100
2/2              0s 55ms/step - loss:
6.3700 - val_loss: 2.4581
Epoch 64/100
2/2              0s 54ms/step - loss:
6.8563 - val_loss: 2.3931
Epoch 65/100
2/2              0s 56ms/step - loss:
6.9176 - val_loss: 2.3314
Epoch 66/100
2/2              0s 55ms/step - loss:
5.0046 - val_loss: 2.3013
Epoch 67/100
2/2              0s 59ms/step - loss:
6.6555 - val_loss: 2.2838
Epoch 68/100
2/2              0s 54ms/step - loss:
```

```
7.0914 - val_loss: 2.2502
Epoch 69/100
2/2              0s 56ms/step - loss:
6.2148 - val_loss: 2.2326
Epoch 70/100
2/2              0s 55ms/step - loss:
4.7450 - val_loss: 2.2206
Epoch 71/100
2/2              0s 57ms/step - loss:
4.7313 - val_loss: 2.2255
Epoch 72/100
2/2              0s 58ms/step - loss:
8.8785 - val_loss: 2.2308
Epoch 73/100
2/2              0s 54ms/step - loss:
8.6147 - val_loss: 2.2475
Epoch 74/100
2/2              0s 49ms/step - loss:
5.3099 - val_loss: 2.2455
Epoch 75/100
2/2              0s 55ms/step - loss:
5.6747 - val_loss: 2.2231
Epoch 76/100
2/2              0s 55ms/step - loss:
6.1843 - val_loss: 2.1900
Epoch 77/100
2/2              0s 60ms/step - loss:
6.4010 - val_loss: 2.1629
Epoch 78/100
2/2              0s 62ms/step - loss:
4.0405 - val_loss: 2.1452
Epoch 79/100
2/2              0s 55ms/step - loss:
5.0519 - val_loss: 2.1256
Epoch 80/100
2/2              0s 56ms/step - loss:
6.3993 - val_loss: 2.1285
Epoch 81/100
2/2              0s 59ms/step - loss:
4.4188 - val_loss: 2.1355
Epoch 82/100
2/2              0s 54ms/step - loss:
4.9286 - val_loss: 2.1372
Epoch 83/100
2/2              0s 58ms/step - loss:
5.7649 - val_loss: 2.1178
Epoch 84/100
2/2              0s 59ms/step - loss:
```

```
5.3273 - val_loss: 2.0534
Epoch 85/100
2/2              0s 49ms/step - loss:
4.9604 - val_loss: 1.9992
Epoch 86/100
2/2              0s 54ms/step - loss:
6.1723 - val_loss: 1.9812
Epoch 87/100
2/2              0s 50ms/step - loss:
4.8077 - val_loss: 1.9821
Epoch 88/100
2/2              0s 54ms/step - loss:
5.3595 - val_loss: 1.9925
Epoch 89/100
2/2              0s 68ms/step - loss:
5.0474 - val_loss: 1.9917
Epoch 90/100
2/2              0s 50ms/step - loss:
5.2085 - val_loss: 1.9808
Epoch 91/100
2/2              0s 54ms/step - loss:
5.2051 - val_loss: 1.9458
Epoch 92/100
2/2              0s 57ms/step - loss:
5.9263 - val_loss: 1.9174
Epoch 93/100
2/2              0s 55ms/step - loss:
4.8932 - val_loss: 1.9045
Epoch 94/100
2/2              0s 54ms/step - loss:
4.3232 - val_loss: 1.9065
Epoch 95/100
2/2              0s 55ms/step - loss:
5.2464 - val_loss: 1.9354
Epoch 96/100
2/2              0s 56ms/step - loss:
4.5374 - val_loss: 1.9676
Epoch 97/100
2/2              0s 58ms/step - loss:
7.3960 - val_loss: 1.9637
Epoch 98/100
2/2              0s 56ms/step - loss:
5.7727 - val_loss: 1.9196
Epoch 99/100
2/2              0s 55ms/step - loss:
3.6213 - val_loss: 1.8706
Epoch 100/100
2/2              0s 58ms/step - loss:
```

```
4.7896 - val_loss: 1.8381
2/2              0s 190ms/step
2/2              0s 15ms/step

Train RMSE: 1.2917543628524752
Test RMSE: 2.605083834938823
```



3.1 LSTM 2 : Metrics Table

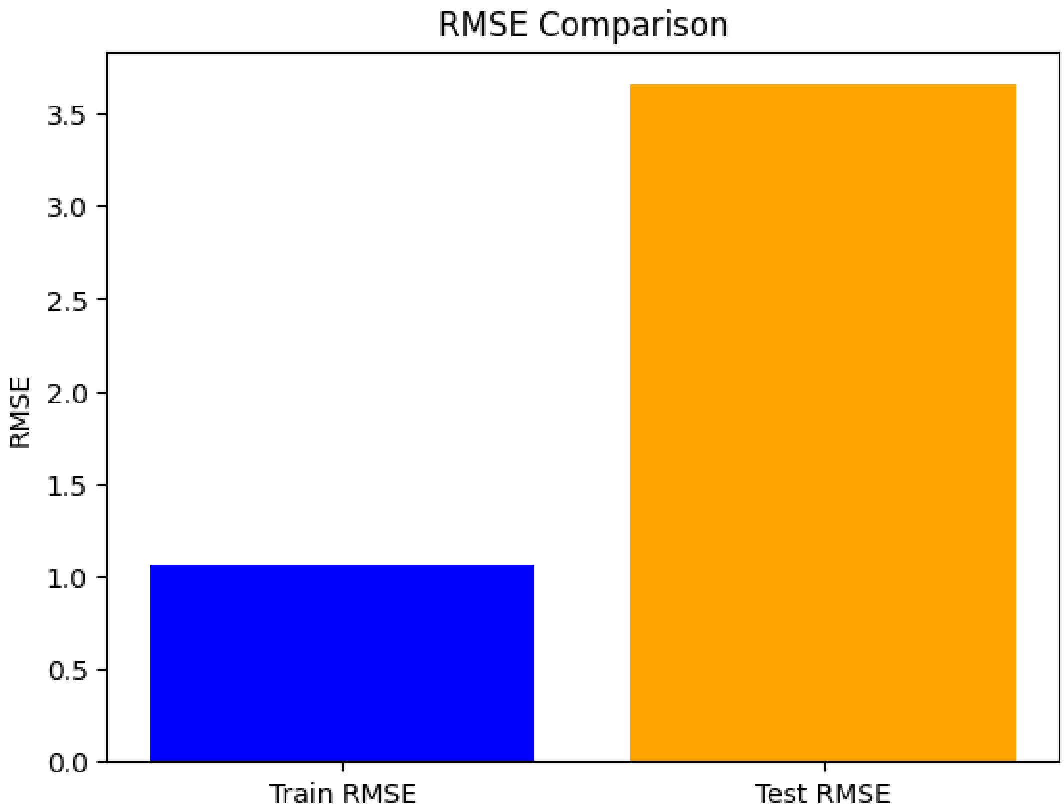
[ ]:
```python
import pandas as pd

# Calculate MSE
mse_train = mean_squared_error(y_train, train_pred)
mse_test = mean_squared_error(y_test, test_pred)

rmse_train = np.sqrt(mse_train)
rmse_test = np.sqrt(mse_test)

# Calculate MAPE
mape_train = np.mean(np.abs((y_train - train_pred.flatten()) / y_train)) * 100
mape_test = np.mean(np.abs((y_test - test_pred.flatten()) / y_test)) * 100
```

```python
# Create a table
results = pd.DataFrame({
    "Metric": ["MSE", "RMSE", "MAPE (%)"],
    "Train": [mse_train, rmse_train, mape_train],
    "Test": [mse_test, rmse_test, mape_test]
})

print("LSTM 4 Model Metrics: Early Stopping\n")
print(results)
```

```
LSTM 4 Model Metrics: Early Stopping

     Metric     Train       Test
0       MSE  1.116986  13.356785
1      RMSE  1.056876   3.654694
2  MAPE (%)  8.784621  27.891186
```

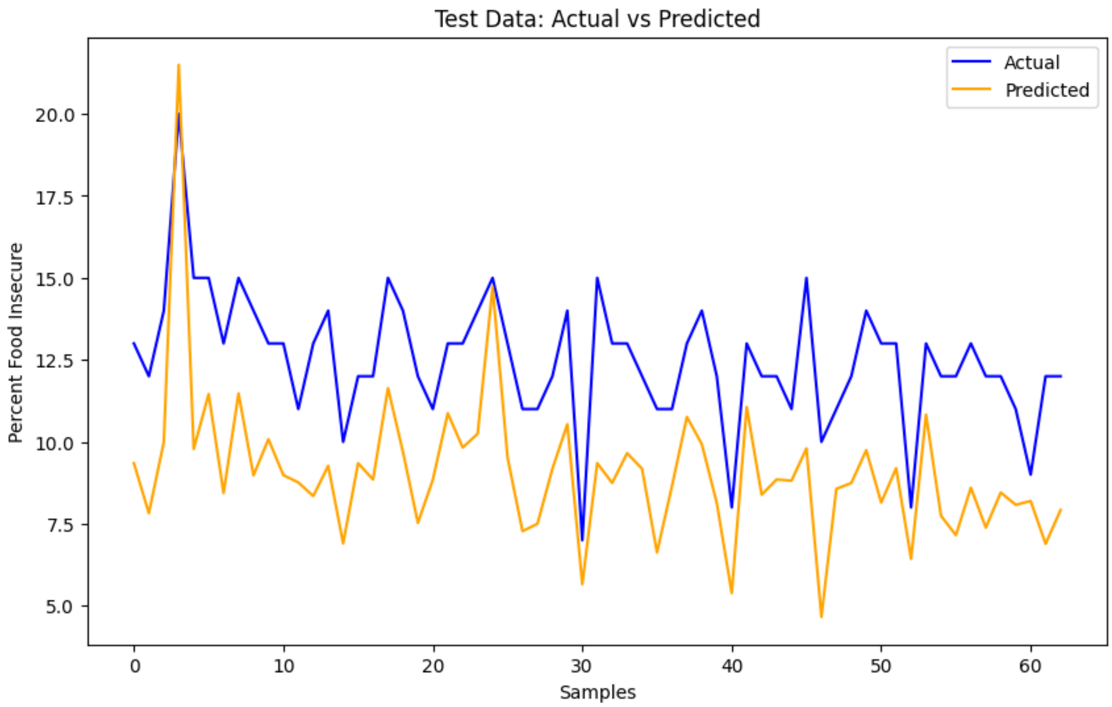
3.2 LSTM 2 : RMSE Training vs Testing

[ ]:
```python
# Visualize RMSE
train_rmse = np.sqrt(mean_squared_error(y_train, train_pred))
test_rmse = np.sqrt(mean_squared_error(y_test, test_pred))

plt.bar(['Train RMSE', 'Test RMSE'], [train_rmse, test_rmse], color=['blue',␣
 ↪'orange'])
plt.title('RMSE Comparison')
plt.ylabel('RMSE')
plt.show()
```



3.3 LSTM 2: Test Data: Predicted vs Actual

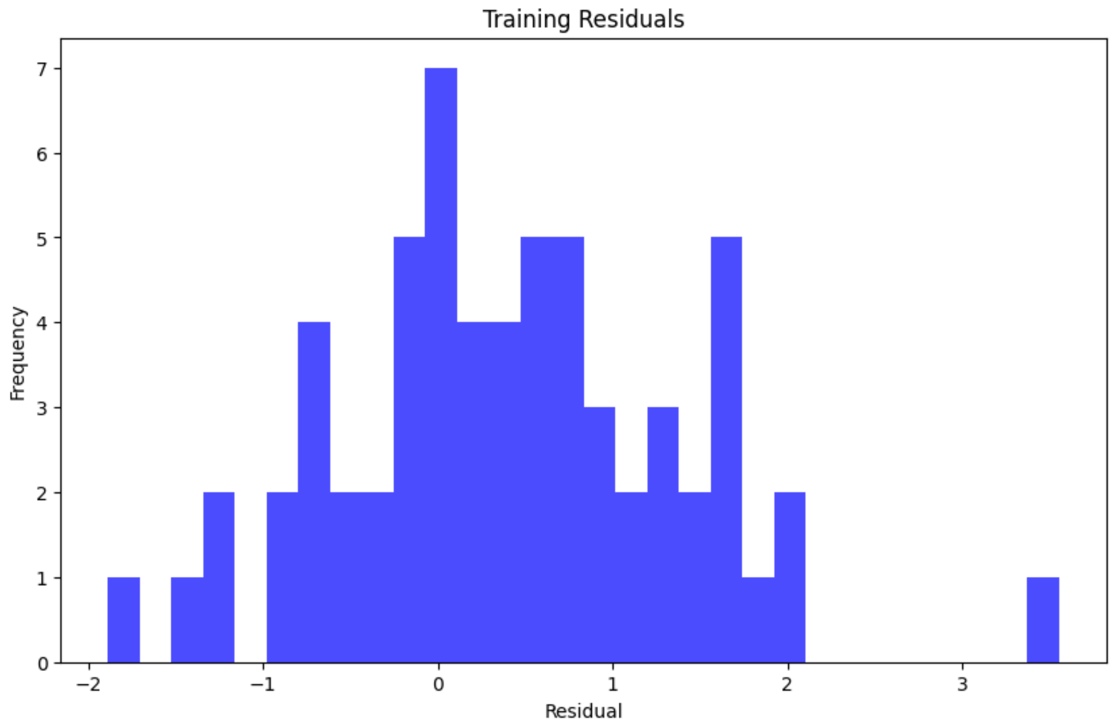
[ ]:
```
plt.figure(figsize=(10, 6))
plt.plot(y_test, label='Actual', color='blue')
plt.plot(test_pred, label='Predicted', color='orange')
plt.title('Test Data: Actual vs Predicted')
plt.xlabel('Samples')
plt.ylabel('Percent Food Insecure')
plt.legend()
plt.show()
```



3.4 Residual Analysis : LSTM 2

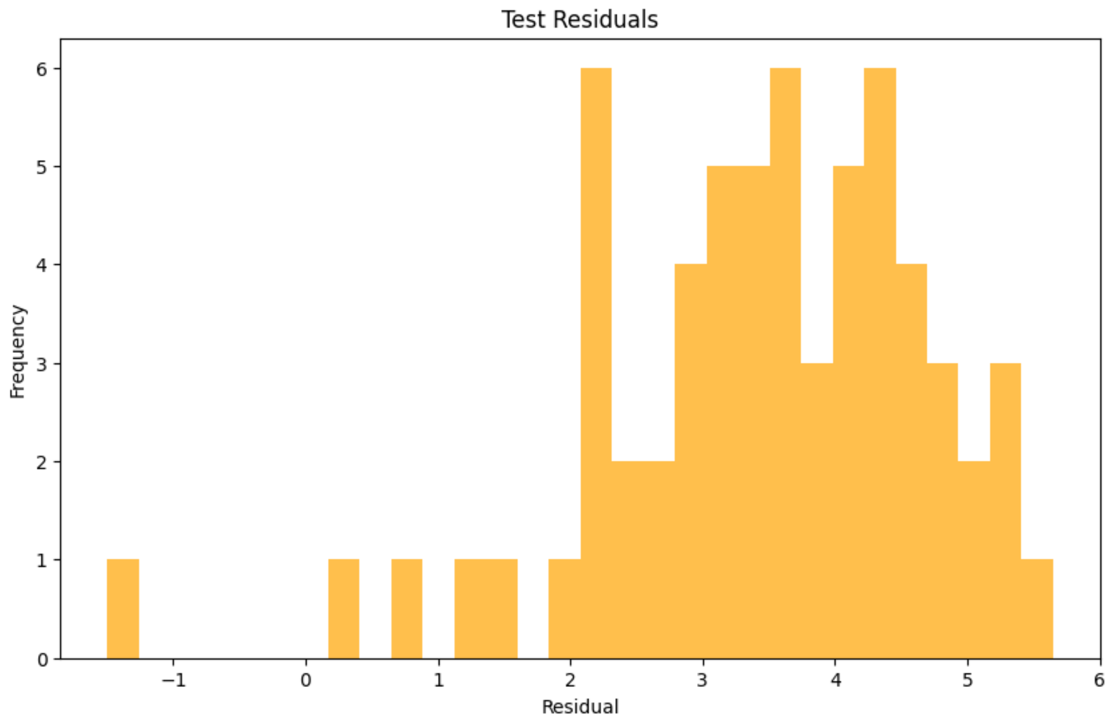
3.4.1 Training

[ ]:
```
train_residuals = y_train - train_pred.flatten()
plt.figure(figsize=(10, 6))
plt.hist(train_residuals, bins=30, color='blue', alpha=0.7)
plt.title('Training Residuals')
plt.xlabel('Residual')
plt.ylabel('Frequency')
plt.show()
```



3.4.2 Testing

[ ]:
```
test_residuals = y_test - test_pred.flatten()
plt.figure(figsize=(10, 6))
plt.hist(test_residuals, bins=30, color='orange', alpha=0.7)
plt.title('Test Residuals')
plt.xlabel('Residual')
plt.ylabel('Frequency')
plt.show()
```



4 LSTM Experiment 3:

- This experiment aims at introducing regularization to the data, l2 regularization from keras will be used here

[ ]:
```
# Step 2: Load data
filepath = "C:\\Users\\jashb\\OneDrive\\Documents\\Masters Data Science\\Spring␣
↪2025\\DATA 698\\Masters Project\\final_data.csv"
df = pd.read_csv(filepath)

print(df.head())
numeric_columns = df.select_dtypes(include=['float64', 'int64']).columns
df[numeric_columns] = df[numeric_columns].fillna(df[numeric_columns].mean())

# Impute categorical columns with the most frequent value
categorical_columns = df.select_dtypes(include=['object']).columns
df[categorical_columns] = df[categorical_columns].
↪fillna(df[categorical_columns].mode().iloc[0])


print(df.isnull().sum())  # Verify no missing values remain

# Step: Drop rows with missing target
```

```python
df = df.dropna(subset=['percent_food_insecure'])

print(df.head())
# Step 3: Drop rows with missing target
df = df.dropna(subset=['percent_food_insecure'])

print(df.head())

# Step 4: Convert 'rural_urban' to numeric
df['rural_urban'] = pd.factorize(df['rural_urban'])[0]

print(df[['rural_urban']].head())

# Step 5: Create lag features
df = df.sort_values(['fips', 'year'])
df['food_insecure_lag1'] = df.groupby('fips')['percent_food_insecure'].shift(1)
df['food_insecure_lag2'] = df.groupby('fips')['percent_food_insecure'].shift(2)

print(df[['fips', 'year', 'percent_food_insecure', 'food_insecure_lag1',␣
 ↪'food_insecure_lag2']].head())

# Step 6: Drop rows with missing lag features
df = df.dropna(subset=['food_insecure_lag1', 'food_insecure_lag2'])

print(df.head())

# Step 7: Select features
features = [
    'percent_household_income_required_for_child_care_expenses',
    'food_environment_index',
    'percent_fair_or_poor_health',
    'percent_unemployed',
    'percent_children_in_poverty',
    'percent_severe_housing_problems',
    'percent_completed_high_school',
    'percent_frequent_mental_distress',
    'percent_uninsured_children',
    'percent_disconnected_youth',
    'spending_per_pupil',
    'school_funding_adequacy',
    'high_school_graduation_rate',
    'median_household_income',
    'gender_pay_gap',
    'percent_enrolled_in_free_or_reduced_lunch',
    'percent_households_with_severe_cost_burden',
    'percent_rural',
    'percent_65_and_over',
```

```python
    'percent_not_proficient_in_english',
    'segregation_index',
    'teen_birth_rate',
    'percent_children_in_single_parent_households',
    'percent_low_birthweight',
    'percent_black',
    'rural_urban',
    'food_insecure_lag1',
    'food_insecure_lag2'
]

available_features = [f for f in features if f in df.columns]
df = df[['year', 'fips', 'county.x', 'state.x', 'percent_food_insecure'] +
 ↪available_features]

print(df.head())

# Step 8: Analyze data

county_years = df.groupby('fips')['year'].count()

# Step 9: Set n_steps
min_years = county_years.min()
n_steps = min(1, min_years)  # Use 3 if possible, otherwise use the minimum
 ↪available
print(f"Using n_steps = {n_steps}")

# Step 10: Split data into train and test
latest_year = df['year'].max()
train = df[df['year'] < 2024]
test = df[df['year'] == 2024]

# Step 11: Prepare training data
counties = train['fips'].unique()
X_train, y_train = [], []
scaler = MinMaxScaler()

all_features = train.drop(columns=['year', 'fips', 'county.x', 'state.x',
 ↪'percent_food_insecure'])
scaler.fit(all_features)

for county in counties:
    county_data = train[train['fips'] == county].sort_values('year')
    if len(county_data) < n_steps:
        continue
    features = county_data.drop(columns=['year', 'fips', 'county.x', 'state.x',
 ↪'percent_food_insecure'])
```

```
    target = county_data['percent_food_insecure'].values
    scaled_features = scaler.transform(features)
    for i in range(n_steps, len(county_data)):
        X_train.append(scaled_features[i-n_steps:i])
        y_train.append(target[i])

X_train = np.array(X_train)
y_train = np.array(y_train)

# Step 12: Prepare test data
X_test, y_test = [], []
test_counties = test['fips'].unique()

for county in test_counties:
    county_data = df[(df['fips'] == county) & (df['year'] <= latest_year)].
 ↪sort_values('year')
    if len(county_data) < n_steps + 1:  # Need n_steps years + target year
        continue
    # Get features from n_steps previous years
    features = county_data.iloc[-(n_steps+1):-1].drop(columns=['year', 'fips',␣
 ↪'county.x', 'state.x', 'percent_food_insecure'])
    target = county_data.iloc[-1]['percent_food_insecure']
    scaled_features = scaler.transform(features)
    X_test.append(scaled_features)
    y_test.append(target)

X_test = np.array(X_test)
y_test = np.array(y_test)

# Step 13: Build LSTM model
input_shape = (X_train.shape[1], X_train.shape[2])
model = Sequential([
    LSTM(50, activation='relu', input_shape=input_shape, return_sequences=True,␣
 ↪kernel_regularizer=L2 (0.01)),
    Dropout(0.25),
    LSTM(50, activation='relu', kernel_regularizer=L2(0.01)),
    Dropout(0.25),
    Dense(1)
])
model.compile(optimizer='adam', loss='mse')

# Step 14: Train the model
history = model.fit(X_train, y_train, epochs=100, batch_size=32,␣
 ↪validation_split=0.2, verbose=1)

# Step 15: Evaluate the model
train_pred = model.predict(X_train)
```

```python
test_pred = model.predict(X_test)
print(f"\nTrain RMSE: {np.sqrt(mean_squared_error(y_train, train_pred))}")
print(f"Test RMSE: {np.sqrt(mean_squared_error(y_test, test_pred))}")

# Step 16: Plot training history
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.legend()
plt.title('Model Training History')
plt.show()
```

```
Step 2: Data loaded
   year   fips   state.x   county.x  \
0  2025  36000  New York      Total
1  2025  36001  New York     Albany
2  2025  36003  New York   Allegany
3  2025  36005  New York      Bronx
4  2025  36007  New York     Broome

   percent_household_income_required_for_child_care_expenses  \
0                                               38.0
1                                               37.0
2                                               43.0
3                                               65.0
4                                               39.0

   food_environment_index  percent_fair_or_poor_health  percent_unemployed  \
0                     8.7                           16                 4.2
1                     8.4                           12                 3.3
2                     8.2                           16                 4.3
3                     7.1                           28                 6.8
4                     7.9                           15                 3.9

   percent_children_in_poverty  percent_severe_housing_problems  …  \
0                           19                               23  …
1                           15                               14  …
2                           17                               12  …
3                           36                               39  …
4                           20                               15  …

   percent_65_and_over  percent_not_proficient_in_english  segregation_index  \
0                 18.6                                  7               0.33
1                 18.7                                  2               0.19
2                 20.9                                  1               0.05
3                 15.3                                 15               0.16
4                 20.7                                  1               0.14

   teen_birth_rate  percent_children_in_single_parent_households  \
```

```
0           10.0                                         26.0
1            8.0                                         27.0
2           10.0                                         19.0
3           17.0                                         52.0
4           12.0                                         25.0

   percent_low_birthweight  percent_black  \
0                      NaN            NaN
1                      NaN            NaN
2                      NaN            NaN
3                      NaN            NaN
4                      NaN            NaN

   percent_children_in_single_parent_households.x  \
0                                             NaN
1                                             NaN
2                                             NaN
3                                             NaN
4                                             NaN

   percent_children_in_single_parent_households.y   rural_urban
0                                             NaN  Mostly Urban
1                                             NaN  Mostly Urban
2                                             NaN  Mostly Rural
3                                             NaN  Mostly Urban
4                                             NaN  Mostly Urban

[5 rows x 33 columns]

Step: Imputed missing values
year                                                      0
fips                                                      0
state.x                                                   0
county.x                                                  0
percent_household_income_required_for_child_care_expenses    0
food_environment_index                                    0
percent_fair_or_poor_health                               0
percent_unemployed                                        0
percent_children_in_poverty                               0
percent_severe_housing_problems                           0
percent_completed_high_school                             0
percent_food_insecure                                     0
percent_frequent_mental_distress                          0
percent_uninsured_children                                0
percent_disconnected_youth                                0
spending_per_pupil                                        0
school_funding_adequacy                                   0
high_school_graduation_rate                               0
```

```
median_household_income                                    0
gender_pay_gap                                             0
percent_enrolled_in_free_or_reduced_lunch                  0
percent_households_with_severe_cost_burden                 0
percent_rural                                              0
percent_65_and_over                                        0
percent_not_proficient_in_english                          0
segregation_index                                          0
teen_birth_rate                                            0
percent_children_in_single_parent_households               0
percent_low_birthweight                                    0
percent_black                                              0
percent_children_in_single_parent_households.x             0
percent_children_in_single_parent_households.y             0
rural_urban                                                0
dtype: int64

Step: Dropped rows with missing 'percent_food_insecure'
   year   fips   state.x  county.x  \
0  2025  36000  New York     Total
1  2025  36001  New York    Albany
2  2025  36003  New York  Allegany
3  2025  36005  New York     Bronx
4  2025  36007  New York    Broome

   percent_household_income_required_for_child_care_expenses  \
0                                               38.0
1                                               37.0
2                                               43.0
3                                               65.0
4                                               39.0

   food_environment_index  percent_fair_or_poor_health  percent_unemployed  \
0                     8.7                           16                 4.2
1                     8.4                           12                 3.3
2                     8.2                           16                 4.3
3                     7.1                           28                 6.8
4                     7.9                           15                 3.9

   percent_children_in_poverty  percent_severe_housing_problems  …  \
0                           19                               23  …
1                           15                               14  …
2                           17                               12  …
3                           36                               39  …
4                           20                               15  …

   percent_65_and_over  percent_not_proficient_in_english  segregation_index  \
0                 18.6                                  7               0.33
```

```
1              18.7                                    2               0.19
2              20.9                                    1               0.05
3              15.3                                   15               0.16
4              20.7                                    1               0.14

   teen_birth_rate  percent_children_in_single_parent_households  \
0             10.0                                          26.0
1              8.0                                          27.0
2             10.0                                          19.0
3             17.0                                          52.0
4             12.0                                          25.0

   percent_low_birthweight  percent_black  \
0                 7.292994       6.134286
1                 7.292994       6.134286
2                 7.292994       6.134286
3                 7.292994       6.134286
4                 7.292994       6.134286

   percent_children_in_single_parent_households.x  \
0                                       22.714286
1                                       22.714286
2                                       22.714286
3                                       22.714286
4                                       22.714286

   percent_children_in_single_parent_households.y   rural_urban
0                                       22.349206  Mostly Urban
1                                       22.349206  Mostly Urban
2                                       22.349206  Mostly Rural
3                                       22.349206  Mostly Urban
4                                       22.349206  Mostly Urban

[5 rows x 33 columns]

Step 3: Dropped rows with missing 'percent_food_insecure'
   year   fips   state.x  county.x  \
0  2025  36000  New York     Total
1  2025  36001  New York    Albany
2  2025  36003  New York  Allegany
3  2025  36005  New York     Bronx
4  2025  36007  New York    Broome

   percent_household_income_required_for_child_care_expenses  \
0                                               38.0
1                                               37.0
2                                               43.0
3                                               65.0
```

```
4                                              39.0

   food_environment_index  percent_fair_or_poor_health  percent_unemployed  \
0                     8.7                           16                 4.2
1                     8.4                           12                 3.3
2                     8.2                           16                 4.3
3                     7.1                           28                 6.8
4                     7.9                           15                 3.9

   percent_children_in_poverty  percent_severe_housing_problems  …  \
0                           19                               23  …
1                           15                               14  …
2                           17                               12  …
3                           36                               39  …
4                           20                               15  …

   percent_65_and_over  percent_not_proficient_in_english  segregation_index  \
0                 18.6                                  7               0.33
1                 18.7                                  2               0.19
2                 20.9                                  1               0.05
3                 15.3                                 15               0.16
4                 20.7                                  1               0.14

   teen_birth_rate  percent_children_in_single_parent_households  \
0             10.0                                          26.0
1              8.0                                          27.0
2             10.0                                          19.0
3             17.0                                          52.0
4             12.0                                          25.0

   percent_low_birthweight  percent_black  \
0                 7.292994       6.134286
1                 7.292994       6.134286
2                 7.292994       6.134286
3                 7.292994       6.134286
4                 7.292994       6.134286

   percent_children_in_single_parent_households.x  \
0                                       22.714286
1                                       22.714286
2                                       22.714286
3                                       22.714286
4                                       22.714286

   percent_children_in_single_parent_households.y   rural_urban
0                                       22.349206  Mostly Urban
1                                       22.349206  Mostly Urban
2                                       22.349206  Mostly Rural
```

```
3                                       22.349206  Mostly Urban
4                                       22.349206  Mostly Urban

[5 rows x 33 columns]

Step 4: Converted 'rural_urban' to numeric
   rural_urban
0            0
1            0
2            1
3            0
4            0

Step 5: Created lag features
      fips  year  percent_food_insecure  food_insecure_lag1  \
315  36000  2020                     11                 NaN
252  36000  2021                     11                11.0
189  36000  2022                     11                11.0
126  36000  2023                     10                11.0
63   36000  2024                     11                10.0

     food_insecure_lag2
315                 NaN
252                 NaN
189                11.0
126                11.0
63                 11.0

Step 6: Dropped rows with missing lag features
     year   fips   state.x county.x  \
189  2022  36000  New York    Total
126  2023  36000  New York    Total
63   2024  36000  New York    Total
0    2025  36000  New York    Total
190  2022  36001  New York   Albany

     percent_household_income_required_for_child_care_expenses  \
189                                           36.26455
126                                           32.00000
63                                            38.00000
0                                             38.00000
190                                           36.26455

     food_environment_index  percent_fair_or_poor_health  percent_unemployed  \
189                     9.0                           16                10.0
126                     8.9                           12                 6.9
63                      8.6                           14                 4.3
0                       8.7                           16                 4.2
```

```
190                    8.3                            15                 7.2

     percent_children_in_poverty  percent_severe_housing_problems  …  \
189                           17                               23  …
126                           19                               23  …
63                            19                               22  …
0                             19                               23  …
190                           13                               15  …

     segregation_index  teen_birth_rate  \
189               0.35             13.0
126               0.34             13.0
63                0.34             11.0
0                 0.33             10.0
190               0.21              9.0

     percent_children_in_single_parent_households  percent_low_birthweight  \
189                                     22.248677                 8.000000
126                                     22.248677                 8.000000
63                                      26.000000                 8.000000
0                                       26.000000                 7.292994
190                                     22.248677                 8.000000

     percent_black  percent_children_in_single_parent_households.x  \
189      14.400000                                       26.000000
126      14.400000                                       26.000000
63       14.400000                                       22.714286
0         6.134286                                       22.714286
190      12.900000                                       29.000000

     percent_children_in_single_parent_households.y  rural_urban  \
189                                       26.000000            0
126                                       26.000000            0
63                                        22.349206            0
0                                         22.349206            0
190                                       27.000000            0

     food_insecure_lag1  food_insecure_lag2
189                11.0                11.0
126                11.0                11.0
63                 10.0                11.0
0                  11.0                10.0
190                10.0                12.0

[5 rows x 35 columns]

Step 7: Selected features
     year   fips county.x   state.x  percent_food_insecure  \
```

```
189  2022  36000    Total  New York                     11
126  2023  36000    Total  New York                     10
63   2024  36000    Total  New York                     11
0    2025  36000    Total  New York                     13
190  2022  36001   Albany  New York                     10

     percent_household_income_required_for_child_care_expenses  \
189                                           36.26455
126                                           32.00000
63                                            38.00000
0                                             38.00000
190                                           36.26455

     food_environment_index  percent_fair_or_poor_health  percent_unemployed  \
189                     9.0                           16                10.0
126                     8.9                           12                 6.9
63                      8.6                           14                 4.3
0                       8.7                           16                 4.2
190                     8.3                           15                 7.2

     percent_children_in_poverty  …  percent_65_and_over  \
189                           17  …                 17.4
126                           19  …                 17.5
63                            19  …                 18.1
0                             19  …                 18.6
190                           13  …                 17.9

     percent_not_proficient_in_english  segregation_index  teen_birth_rate  \
189                                  7               0.35             13.0
126                                  7               0.34             13.0
63                                   7               0.34             11.0
0                                    7               0.33             10.0
190                                  2               0.21              9.0

     percent_children_in_single_parent_households  percent_low_birthweight  \
189                                     22.248677                 8.000000
126                                     22.248677                 8.000000
63                                      26.000000                 8.000000
0                                       26.000000                 7.292994
190                                     22.248677                 8.000000

     percent_black  rural_urban  food_insecure_lag1  food_insecure_lag2
189      14.400000            0                11.0                11.0
126      14.400000            0                11.0                11.0
63       14.400000            0                10.0                11.0
0         6.134286            0                11.0                10.0
190      12.900000            0                10.0                12.0
```

```
[5 rows x 33 columns]

Step 8: Data Analysis
Years available: [2022, 2023, 2024, 2025]
Counties with data: 63

Minimum years per county: 4
Maximum years per county: 4
Using n_steps = 1

Training years: [2022, 2023]
Test year: [2024]

Training data shape: (63, 1, 28)
Training target shape: (63,)

Test data shape: (63, 1, 28)
Test target shape: (63,)
Epoch 1/100

c:\Users\jashb\Lib\site-packages\keras\src\layers\rnn\rnn.py:200: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When using
Sequential models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(**kwargs)

2/2            3s 329ms/step - loss:
142.5095 - val_loss: 109.5390
Epoch 2/100
2/2            0s 52ms/step - loss:
141.7531 - val_loss: 109.2682
Epoch 3/100
2/2            0s 50ms/step - loss:
142.8644 - val_loss: 109.0007
Epoch 4/100
2/2            0s 52ms/step - loss:
140.2984 - val_loss: 108.7337
Epoch 5/100
2/2            0s 56ms/step - loss:
139.3163 - val_loss: 108.4632
Epoch 6/100
2/2            0s 63ms/step - loss:
138.6150 - val_loss: 108.1787
Epoch 7/100
2/2            0s 51ms/step - loss:
140.9854 - val_loss: 107.8723
Epoch 8/100
2/2            0s 52ms/step - loss:
136.6801 - val_loss: 107.5365
```

```
Epoch 9/100
2/2             0s 49ms/step - loss:
139.4378 - val_loss: 107.1662
Epoch 10/100
2/2             0s 51ms/step - loss:
139.3555 - val_loss: 106.7513
Epoch 11/100
2/2             0s 51ms/step - loss:
141.8414 - val_loss: 106.2836
Epoch 12/100
2/2             0s 51ms/step - loss:
134.4766 - val_loss: 105.7576
Epoch 13/100
2/2             0s 54ms/step - loss:
135.5411 - val_loss: 105.1641
Epoch 14/100
2/2             0s 50ms/step - loss:
133.2112 - val_loss: 104.4868
Epoch 15/100
2/2             0s 53ms/step - loss:
132.7950 - val_loss: 103.7070
Epoch 16/100
2/2             0s 50ms/step - loss:
134.9705 - val_loss: 102.8102
Epoch 17/100
2/2             0s 52ms/step - loss:
136.3674 - val_loss: 101.7780
Epoch 18/100
2/2             0s 51ms/step - loss:
135.1101 - val_loss: 100.5851
Epoch 19/100
2/2             0s 55ms/step - loss:
132.2493 - val_loss: 99.2071
Epoch 20/100
2/2             0s 51ms/step - loss:
128.5256 - val_loss: 97.6050
Epoch 21/100
2/2             0s 53ms/step - loss:
128.4224 - val_loss: 95.7351
Epoch 22/100
2/2             0s 53ms/step - loss:
123.7517 - val_loss: 93.5452
Epoch 23/100
2/2             0s 48ms/step - loss:
120.1770 - val_loss: 90.9863
Epoch 24/100
2/2             0s 50ms/step - loss:
115.2840 - val_loss: 87.9892
```

```
Epoch 25/100
2/2             0s 53ms/step - loss:
112.7853 - val_loss: 84.4809
Epoch 26/100
2/2             0s 56ms/step - loss:
109.2631 - val_loss: 80.3854
Epoch 27/100
2/2             0s 50ms/step - loss:
100.1322 - val_loss: 75.6252
Epoch 28/100
2/2             0s 49ms/step - loss:
100.3384 - val_loss: 70.1441
Epoch 29/100
2/2             0s 51ms/step - loss:
94.1933 - val_loss: 63.9111
Epoch 30/100
2/2             0s 50ms/step - loss:
85.6443 - val_loss: 56.9188
Epoch 31/100
2/2             0s 52ms/step - loss:
74.1762 - val_loss: 49.2020
Epoch 32/100
2/2             0s 53ms/step - loss:
68.4294 - val_loss: 40.9044
Epoch 33/100
2/2             0s 51ms/step - loss:
54.7267 - val_loss: 32.2741
Epoch 34/100
2/2             0s 51ms/step - loss:
42.4141 - val_loss: 23.7375
Epoch 35/100
2/2             0s 51ms/step - loss:
34.9354 - val_loss: 15.8421
Epoch 36/100
2/2             0s 52ms/step - loss:
23.3733 - val_loss: 9.3634
Epoch 37/100
2/2             0s 49ms/step - loss:
15.7103 - val_loss: 5.0733
Epoch 38/100
2/2             0s 53ms/step - loss:
9.0135 - val_loss: 3.4661
Epoch 39/100
2/2             0s 50ms/step - loss:
10.8749 - val_loss: 4.2837
Epoch 40/100
2/2             0s 61ms/step - loss:
10.4483 - val_loss: 6.2485
```

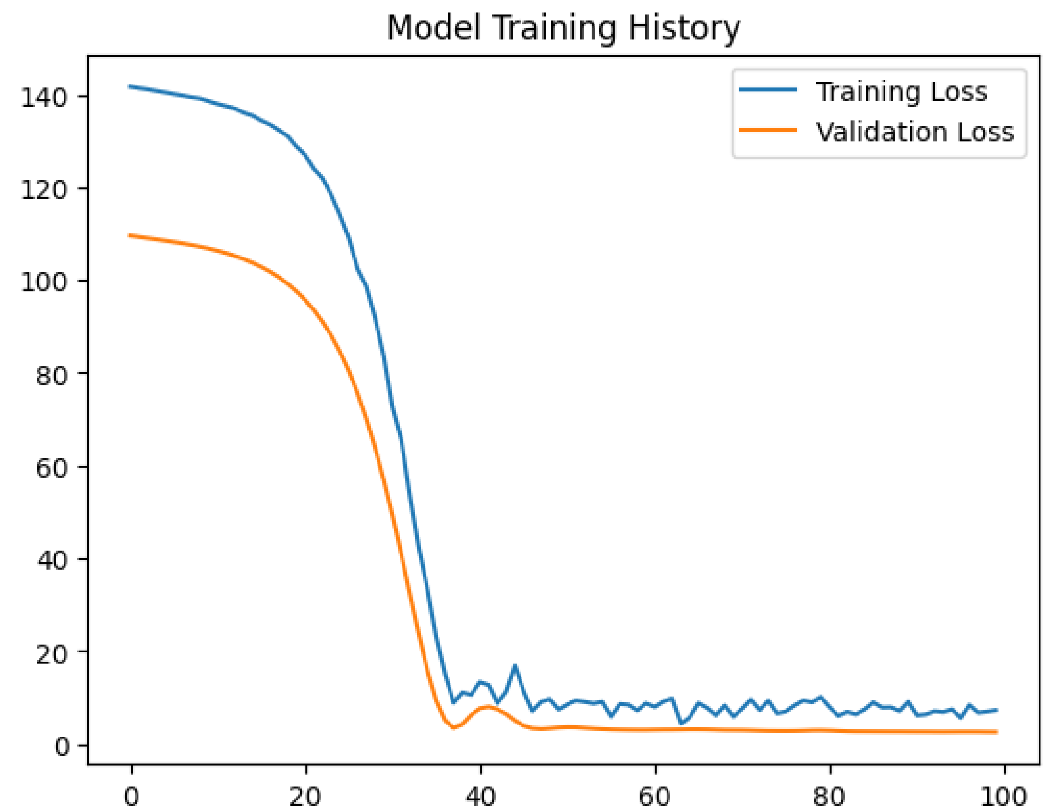
```
Epoch 41/100
2/2              0s 64ms/step - loss:
13.7514 - val_loss: 7.6665
Epoch 42/100
2/2              0s 61ms/step - loss:
12.3664 - val_loss: 8.0232
Epoch 43/100
2/2              0s 55ms/step - loss:
9.4522 - val_loss: 7.4837
Epoch 44/100
2/2              0s 51ms/step - loss:
11.6293 - val_loss: 6.4752
Epoch 45/100
2/2              0s 53ms/step - loss:
18.1763 - val_loss: 5.0316
Epoch 46/100
2/2              0s 55ms/step - loss:
11.5383 - val_loss: 3.9604
Epoch 47/100
2/2              0s 62ms/step - loss:
7.1907 - val_loss: 3.3945
Epoch 48/100
2/2              0s 60ms/step - loss:
9.4649 - val_loss: 3.2593
Epoch 49/100
2/2              0s 53ms/step - loss:
10.0521 - val_loss: 3.3781
Epoch 50/100
2/2              0s 59ms/step - loss:
7.7109 - val_loss: 3.5641
Epoch 51/100
2/2              0s 53ms/step - loss:
8.9765 - val_loss: 3.6488
Epoch 52/100
2/2              0s 55ms/step - loss:
9.3741 - val_loss: 3.6217
Epoch 53/100
2/2              0s 54ms/step - loss:
8.5674 - val_loss: 3.4763
Epoch 54/100
2/2              0s 58ms/step - loss:
8.5837 - val_loss: 3.3358
Epoch 55/100
2/2              0s 67ms/step - loss:
9.1631 - val_loss: 3.2425
Epoch 56/100
2/2              0s 56ms/step - loss:
6.2229 - val_loss: 3.1609
```

```
Epoch 57/100
2/2            0s 53ms/step - loss:
8.0564 - val_loss: 3.1157
Epoch 58/100
2/2            0s 55ms/step - loss:
7.9190 - val_loss: 3.0841
Epoch 59/100
2/2            0s 57ms/step - loss:
7.5194 - val_loss: 3.0648
Epoch 60/100
2/2            0s 54ms/step - loss:
9.0599 - val_loss: 3.0643
Epoch 61/100
2/2            0s 53ms/step - loss:
7.7066 - val_loss: 3.0966
Epoch 62/100
2/2            0s 50ms/step - loss:
9.4903 - val_loss: 3.1244
Epoch 63/100
2/2            0s 60ms/step - loss:
9.8230 - val_loss: 3.1291
Epoch 64/100
2/2            0s 50ms/step - loss:
4.2070 - val_loss: 3.1595
Epoch 65/100
2/2            0s 54ms/step - loss:
5.6823 - val_loss: 3.1937
Epoch 66/100
2/2            0s 54ms/step - loss:
8.9027 - val_loss: 3.1895
Epoch 67/100
2/2            0s 55ms/step - loss:
7.7238 - val_loss: 3.1618
Epoch 68/100
2/2            0s 48ms/step - loss:
5.7825 - val_loss: 3.1035
Epoch 69/100
2/2            0s 55ms/step - loss:
8.3028 - val_loss: 3.0436
Epoch 70/100
2/2            0s 55ms/step - loss:
6.1496 - val_loss: 3.0249
Epoch 71/100
2/2            0s 54ms/step - loss:
7.9921 - val_loss: 3.0143
Epoch 72/100
2/2            0s 54ms/step - loss:
9.7205 - val_loss: 2.9672
```

```
Epoch 73/100
2/2            0s 52ms/step - loss:
7.4715 - val_loss: 2.9062
Epoch 74/100
2/2            0s 54ms/step - loss:
9.3319 - val_loss: 2.8638
Epoch 75/100
2/2            0s 60ms/step - loss:
7.1135 - val_loss: 2.8452
Epoch 76/100
2/2            0s 53ms/step - loss:
7.0794 - val_loss: 2.8361
Epoch 77/100
2/2            0s 55ms/step - loss:
8.2369 - val_loss: 2.8487
Epoch 78/100
2/2            0s 49ms/step - loss:
8.7710 - val_loss: 2.8950
Epoch 79/100
2/2            0s 55ms/step - loss:
9.2587 - val_loss: 2.9608
Epoch 80/100
2/2            0s 52ms/step - loss:
10.2348 - val_loss: 2.9775
Epoch 81/100
2/2            0s 55ms/step - loss:
8.3001 - val_loss: 2.9235
Epoch 82/100
2/2            0s 59ms/step - loss:
5.5440 - val_loss: 2.8427
Epoch 83/100
2/2            0s 56ms/step - loss:
7.0101 - val_loss: 2.7717
Epoch 84/100
2/2            0s 53ms/step - loss:
6.5503 - val_loss: 2.7392
Epoch 85/100
2/2            0s 53ms/step - loss:
7.3465 - val_loss: 2.7306
Epoch 86/100
2/2            0s 51ms/step - loss:
8.9530 - val_loss: 2.7194
Epoch 87/100
2/2            0s 53ms/step - loss:
7.5950 - val_loss: 2.7090
Epoch 88/100
2/2            0s 54ms/step - loss:
8.2358 - val_loss: 2.7000
```

```
Epoch 89/100
2/2            0s 55ms/step - loss:
7.0824 - val_loss: 2.6955
Epoch 90/100
2/2            0s 55ms/step - loss:
9.2040 - val_loss: 2.6833
Epoch 91/100
2/2            0s 54ms/step - loss:
6.0206 - val_loss: 2.6738
Epoch 92/100
2/2            0s 53ms/step - loss:
6.5621 - val_loss: 2.6678
Epoch 93/100
2/2            0s 55ms/step - loss:
7.0872 - val_loss: 2.6539
Epoch 94/100
2/2            0s 52ms/step - loss:
6.7507 - val_loss: 2.6441
Epoch 95/100
2/2            0s 55ms/step - loss:
7.8044 - val_loss: 2.6528
Epoch 96/100
2/2            0s 53ms/step - loss:
5.7362 - val_loss: 2.6634
Epoch 97/100
2/2            0s 53ms/step - loss:
7.6787 - val_loss: 2.6617
Epoch 98/100
2/2            0s 55ms/step - loss:
6.7088 - val_loss: 2.6554
Epoch 99/100
2/2            0s 57ms/step - loss:
7.3367 - val_loss: 2.6291
Epoch 100/100
2/2            0s 61ms/step - loss:
7.6919 - val_loss: 2.6101
WARNING:tensorflow:5 out of the last 9 calls to <function
TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_distributed at
0x0000020235D20CC0> triggered tf.function retracing. Tracing is expensive and
the excessive number of tracings could be due to (1) creating @tf.function
repeatedly in a loop, (2) passing tensors with different shapes, (3) passing
Python objects instead of tensors. For (1), please define your @tf.function
outside of the loop. For (2), @tf.function has reduce_retracing=True option that
can avoid unnecessary retracing. For (3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.
1/2            0s
179ms/stepWARNING:tensorflow:6 out of the last 10 calls to <function
```

```
TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_distributed at
0x0000020235D20CC0> triggered tf.function retracing. Tracing is expensive and
the excessive number of tracings could be due to (1) creating @tf.function
repeatedly in a loop, (2) passing tensors with different shapes, (3) passing
Python objects instead of tensors. For (1), please define your @tf.function
outside of the loop. For (2), @tf.function has reduce_retracing=True option that
can avoid unnecessary retracing. For (3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.
2/2              0s 187ms/step
2/2              0s 13ms/step

Train RMSE: 1.329116826174556
Test RMSE: 2.090067890633188
```



4.1 LSTM 3 : Metrics Table

- MAPE and MSE

[ ]:
```
import pandas as pd
```

```
# Calculate MSE
mse_train = mean_squared_error(y_train, train_pred)
mse_test = mean_squared_error(y_test, test_pred)

rmse_train = np.sqrt(mse_train)
rmse_test = np.sqrt(mse_test)

# Calculate MAPE
mape_train = np.mean(np.abs((y_train - train_pred.flatten()) / y_train)) * 100
mape_test = np.mean(np.abs((y_test - test_pred.flatten()) / y_test)) * 100

# Create a table
results = pd.DataFrame({
    "Metric": ["MSE", "RMSE", "MAPE (%)"],
    "Train": [mse_train, rmse_train, mape_train],
    "Test": [mse_test, rmse_test, mape_test]
})

print("LSTM 4 Model Metrics: Early Stopping\n")
print(results)
```

```
LSTM 4 Model Metrics: Early Stopping

     Metric     Train       Test
0       MSE  1.185769   4.735223
1      RMSE  1.088930   2.176057
2  MAPE (%)  8.900780  15.370359
```

5 LSTM Experiment 4: (Best Performing)

- This experiment will be introducing early stopping to try and limit the overfitting in the model

[ ]:
```
# Step 2: Load data
filepath = "C:\\Users\\jashb\\OneDrive\\Documents\\Masters Data Science\\Spring␣
↪2025\\DATA 698\\Masters Project\\final_data.csv"
df = pd.read_csv(filepath)

print(df.head())
# Step: Impute missing values
# Impute numeric columns with their mean
numeric_columns = df.select_dtypes(include=['float64', 'int64']).columns
df[numeric_columns] = df[numeric_columns].fillna(df[numeric_columns].mean())

# Impute categorical columns with the most frequent value
categorical_columns = df.select_dtypes(include=['object']).columns
```

```python
df[categorical_columns] = df[categorical_columns].
 ↪fillna(df[categorical_columns].mode().iloc[0])


print(df.isnull().sum())  # Verify no missing values remain

# Step: Drop rows with missing target
df = df.dropna(subset=['percent_food_insecure'])

print(df.head())
# Step 3: Drop rows with missing target
df = df.dropna(subset=['percent_food_insecure'])

print(df.head())

# Step 4: Convert 'rural_urban' to numeric
df['rural_urban'] = pd.factorize(df['rural_urban'])[0]

print(df[['rural_urban']].head())

# Step 5: Create lag features
df = df.sort_values(['fips', 'year'])
df['food_insecure_lag1'] = df.groupby('fips')['percent_food_insecure'].shift(1)
df['food_insecure_lag2'] = df.groupby('fips')['percent_food_insecure'].shift(2)

print(df[['fips', 'year', 'percent_food_insecure', 'food_insecure_lag1',␣
 ↪'food_insecure_lag2']].head())

# Step 6: Drop rows with missing lag features
df = df.dropna(subset=['food_insecure_lag1', 'food_insecure_lag2'])

print(df.head())

# Step 7: Select features
features = [
    'percent_household_income_required_for_child_care_expenses',
    'food_environment_index',
    'percent_fair_or_poor_health',
    'percent_unemployed',
    'percent_children_in_poverty',
    'percent_severe_housing_problems',
    'percent_completed_high_school',
    'percent_frequent_mental_distress',
    'percent_uninsured_children',
    'percent_disconnected_youth',
    'spending_per_pupil',
    'school_funding_adequacy',
```

```
    'high_school_graduation_rate',
    'median_household_income',
    'gender_pay_gap',
    'percent_enrolled_in_free_or_reduced_lunch',
    'percent_households_with_severe_cost_burden',
    'percent_rural',
    'percent_65_and_over',
    'percent_not_proficient_in_english',
    'segregation_index',
    'teen_birth_rate',
    'percent_children_in_single_parent_households',
    'percent_low_birthweight',
    'percent_black',
    'rural_urban',
    'food_insecure_lag1',
    'food_insecure_lag2'
]

available_features = [f for f in features if f in df.columns]
df = df[['year', 'fips', 'county.x', 'state.x', 'percent_food_insecure'] +␣
 ↪available_features]

print(df.head())

county_years = df.groupby('fips')['year'].count()

# Step 9: Set n_steps
min_years = county_years.min()
n_steps = min(1, min_years)
print(f"Using n_steps = {n_steps}")

# Step 10: Split data into train and test
latest_year = df['year'].max()
train = df[df['year'] < 2024]
test = df[df['year'] == 2024]

# Step 11: Prepare training data
counties = train['fips'].unique()
X_train, y_train = [], []
scaler = MinMaxScaler()

all_features = train.drop(columns=['year', 'fips', 'county.x', 'state.x',␣
 ↪'percent_food_insecure'])
scaler.fit(all_features)

for county in counties:
    county_data = train[train['fips'] == county].sort_values('year')
```

```
    if len(county_data) < n_steps:
        continue
    features = county_data.drop(columns=['year', 'fips', 'county.x', 'state.x', 'percent_food_insecure'])
    target = county_data['percent_food_insecure'].values
    scaled_features = scaler.transform(features)
    for i in range(n_steps, len(county_data)):
        X_train.append(scaled_features[i-n_steps:i])
        y_train.append(target[i])

X_train = np.array(X_train)
y_train = np.array(y_train)
# Step 12: Prepare test data
X_test, y_test = [], []
test_counties = test['fips'].unique()

for county in test_counties:
    county_data = df[(df['fips'] == county) & (df['year'] <= latest_year)].sort_values('year')
    if len(county_data) < n_steps + 1:  # Need n_steps years + target year
        continue
    # Get features from n_steps previous years
    features = county_data.iloc[-(n_steps+1):-1].drop(columns=['year', 'fips', 'county.x', 'state.x', 'percent_food_insecure'])
    target = county_data.iloc[-1]['percent_food_insecure']
    scaled_features = scaler.transform(features)
    X_test.append(scaled_features)
    y_test.append(target)

X_test = np.array(X_test)
y_test = np.array(y_test)


# Step 13: Build LSTM model
input_shape = (X_train.shape[1], X_train.shape[2])
model = Sequential([
    LSTM(50, activation='relu', input_shape=input_shape, return_sequences=True, kernel_regularizer=L2 (0.01)),
    Dropout(0.15),
    LSTM(50, activation='relu', kernel_regularizer=L2(0.01)),
    Dropout(0.15),
    Dense(1)
])
model.compile(optimizer='adam', loss='mse')

# An early stopping callback to attempt to prevent overfitting
early_stopping = EarlyStopping(
```

```
    monitor='val_loss',   # Monitor validation loss
    patience=15,          # Stop training after 10 epochs with no improvement
    restore_best_weights=True  # Restore the best weights after stopping
)
# Step 14: Train the model
history = model.fit(
    X_train, y_train,
    epochs=100,
    batch_size=32,
    validation_split=0.2,
    verbose=1,
    callbacks=[early_stopping]  # Add the callback here
)

# Step 15: Evaluate the model
train_pred = model.predict(X_train)
test_pred = model.predict(X_test)

# Step 16: Plot training history
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.legend()
plt.title('Model Training History')
plt.show()
```

```
Step 2: Data loaded
   year   fips   state.x  county.x  \
0  2025  36000  New York     Total
1  2025  36001  New York    Albany
2  2025  36003  New York  Allegany
3  2025  36005  New York     Bronx
4  2025  36007  New York    Broome

   percent_household_income_required_for_child_care_expenses  \
0                                               38.0
1                                               37.0
2                                               43.0
3                                               65.0
4                                               39.0

   food_environment_index  percent_fair_or_poor_health  percent_unemployed  \
0                     8.7                           16                 4.2
1                     8.4                           12                 3.3
2                     8.2                           16                 4.3
3                     7.1                           28                 6.8
4                     7.9                           15                 3.9

   percent_children_in_poverty  percent_severe_housing_problems  …  \
```

```
0                          19                                23  …
1                          15                                14  …
2                          17                                12  …
3                          36                                39  …
4                          20                                15  …

   percent_65_and_over  percent_not_proficient_in_english  segregation_index  \
0                 18.6                                  7               0.33
1                 18.7                                  2               0.19
2                 20.9                                  1               0.05
3                 15.3                                 15               0.16
4                 20.7                                  1               0.14

   teen_birth_rate  percent_children_in_single_parent_households  \
0             10.0                                          26.0
1              8.0                                          27.0
2             10.0                                          19.0
3             17.0                                          52.0
4             12.0                                          25.0

   percent_low_birthweight  percent_black  \
0                      NaN            NaN
1                      NaN            NaN
2                      NaN            NaN
3                      NaN            NaN
4                      NaN            NaN

   percent_children_in_single_parent_households.x  \
0                                             NaN
1                                             NaN
2                                             NaN
3                                             NaN
4                                             NaN

   percent_children_in_single_parent_households.y   rural_urban
0                                             NaN  Mostly Urban
1                                             NaN  Mostly Urban
2                                             NaN  Mostly Rural
3                                             NaN  Mostly Urban
4                                             NaN  Mostly Urban

[5 rows x 33 columns]

Step: Imputed missing values
year                                                            0
fips                                                            0
state.x                                                         0
county.x                                                        0
```

```
percent_household_income_required_for_child_care_expenses    0
food_environment_index                                       0
percent_fair_or_poor_health                                  0
percent_unemployed                                           0
percent_children_in_poverty                                  0
percent_severe_housing_problems                              0
percent_completed_high_school                                0
percent_food_insecure                                        0
percent_frequent_mental_distress                             0
percent_uninsured_children                                   0
percent_disconnected_youth                                   0
spending_per_pupil                                           0
school_funding_adequacy                                      0
high_school_graduation_rate                                  0
median_household_income                                      0
gender_pay_gap                                               0
percent_enrolled_in_free_or_reduced_lunch                    0
percent_households_with_severe_cost_burden                   0
percent_rural                                                0
percent_65_and_over                                          0
percent_not_proficient_in_english                            0
segregation_index                                            0
teen_birth_rate                                              0
percent_children_in_single_parent_households                 0
percent_low_birthweight                                      0
percent_black                                                0
percent_children_in_single_parent_households.x               0
percent_children_in_single_parent_households.y               0
rural_urban                                                  0
dtype: int64

Step: Dropped rows with missing 'percent_food_insecure'
   year   fips   state.x  county.x  \
0  2025  36000  New York     Total
1  2025  36001  New York    Albany
2  2025  36003  New York  Allegany
3  2025  36005  New York     Bronx
4  2025  36007  New York    Broome

   percent_household_income_required_for_child_care_expenses  \
0                                               38.0
1                                               37.0
2                                               43.0
3                                               65.0
4                                               39.0

   food_environment_index  percent_fair_or_poor_health  percent_unemployed  \
0                    8.7                           16                 4.2
```

```
1                  8.4                   12              3.3
2                  8.2                   16              4.3
3                  7.1                   28              6.8
4                  7.9                   15              3.9

   percent_children_in_poverty  percent_severe_housing_problems  …  \
0                           19                               23  …
1                           15                               14  …
2                           17                               12  …
3                           36                               39  …
4                           20                               15  …

   percent_65_and_over  percent_not_proficient_in_english  segregation_index  \
0                 18.6                                  7               0.33
1                 18.7                                  2               0.19
2                 20.9                                  1               0.05
3                 15.3                                 15               0.16
4                 20.7                                  1               0.14

   teen_birth_rate  percent_children_in_single_parent_households  \
0             10.0                                          26.0
1              8.0                                          27.0
2             10.0                                          19.0
3             17.0                                          52.0
4             12.0                                          25.0

   percent_low_birthweight  percent_black  \
0                 7.292994       6.134286
1                 7.292994       6.134286
2                 7.292994       6.134286
3                 7.292994       6.134286
4                 7.292994       6.134286

   percent_children_in_single_parent_households.x  \
0                                       22.714286
1                                       22.714286
2                                       22.714286
3                                       22.714286
4                                       22.714286

   percent_children_in_single_parent_households.y   rural_urban
0                                       22.349206  Mostly Urban
1                                       22.349206  Mostly Urban
2                                       22.349206  Mostly Rural
3                                       22.349206  Mostly Urban
4                                       22.349206  Mostly Urban

[5 rows x 33 columns]
```

```
Step 3: Dropped rows with missing 'percent_food_insecure'
   year   fips   state.x  county.x  \
0  2025  36000  New York     Total
1  2025  36001  New York    Albany
2  2025  36003  New York  Allegany
3  2025  36005  New York     Bronx
4  2025  36007  New York    Broome

   percent_household_income_required_for_child_care_expenses  \
0                                               38.0
1                                               37.0
2                                               43.0
3                                               65.0
4                                               39.0

   food_environment_index  percent_fair_or_poor_health  percent_unemployed  \
0                     8.7                           16                 4.2
1                     8.4                           12                 3.3
2                     8.2                           16                 4.3
3                     7.1                           28                 6.8
4                     7.9                           15                 3.9

   percent_children_in_poverty  percent_severe_housing_problems  …  \
0                           19                               23  …
1                           15                               14  …
2                           17                               12  …
3                           36                               39  …
4                           20                               15  …

   percent_65_and_over  percent_not_proficient_in_english  segregation_index  \
0                 18.6                                  7               0.33
1                 18.7                                  2               0.19
2                 20.9                                  1               0.05
3                 15.3                                 15               0.16
4                 20.7                                  1               0.14

   teen_birth_rate  percent_children_in_single_parent_households  \
0             10.0                                          26.0
1              8.0                                          27.0
2             10.0                                          19.0
3             17.0                                          52.0
4             12.0                                          25.0

   percent_low_birthweight  percent_black  \
0                 7.292994       6.134286
1                 7.292994       6.134286
2                 7.292994       6.134286
```

```
3                 7.292994       6.134286
4                 7.292994       6.134286

   percent_children_in_single_parent_households.x  \
0                                        22.714286
1                                        22.714286
2                                        22.714286
3                                        22.714286
4                                        22.714286

   percent_children_in_single_parent_households.y   rural_urban
0                                        22.349206  Mostly Urban
1                                        22.349206  Mostly Urban
2                                        22.349206  Mostly Rural
3                                        22.349206  Mostly Urban
4                                        22.349206  Mostly Urban

[5 rows x 33 columns]

Step 4: Converted 'rural_urban' to numeric
   rural_urban
0            0
1            0
2            1
3            0
4            0

Step 5: Created lag features
      fips  year  percent_food_insecure  food_insecure_lag1  \
315  36000  2020                     11                 NaN
252  36000  2021                     11                11.0
189  36000  2022                     11                11.0
126  36000  2023                     10                11.0
63   36000  2024                     11                10.0

     food_insecure_lag2
315                 NaN
252                 NaN
189                11.0
126                11.0
63                 11.0

Step 6: Dropped rows with missing lag features
     year   fips   state.x county.x  \
189  2022  36000  New York    Total
126  2023  36000  New York    Total
63   2024  36000  New York    Total
0    2025  36000  New York    Total
```

```
190  2022  36001  New York   Albany

     percent_household_income_required_for_child_care_expenses  \
189                                           36.26455
126                                           32.00000
63                                            38.00000
0                                             38.00000
190                                           36.26455

     food_environment_index  percent_fair_or_poor_health  percent_unemployed  \
189                     9.0                           16                10.0
126                     8.9                           12                 6.9
63                      8.6                           14                 4.3
0                       8.7                           16                 4.2
190                     8.3                           15                 7.2

     percent_children_in_poverty  percent_severe_housing_problems  …  \
189                           17                               23  …
126                           19                               23  …
63                            19                               22  …
0                             19                               23  …
190                           13                               15  …

     segregation_index  teen_birth_rate  \
189               0.35             13.0
126               0.34             13.0
63                0.34             11.0
0                 0.33             10.0
190               0.21              9.0

     percent_children_in_single_parent_households  percent_low_birthweight  \
189                                     22.248677                 8.000000
126                                     22.248677                 8.000000
63                                      26.000000                 8.000000
0                                       26.000000                 7.292994
190                                     22.248677                 8.000000

     percent_black  percent_children_in_single_parent_households.x  \
189      14.400000                                       26.000000
126      14.400000                                       26.000000
63       14.400000                                       22.714286
0         6.134286                                       22.714286
190      12.900000                                       29.000000

     percent_children_in_single_parent_households.y  rural_urban  \
189                                       26.000000            0
126                                       26.000000            0
63                                        22.349206            0
```

```
0                                         22.349206             0
190                                       27.000000             0

     food_insecure_lag1  food_insecure_lag2
189                11.0                11.0
126                11.0                11.0
63                 10.0                11.0
0                  11.0                10.0
190                10.0                12.0

[5 rows x 35 columns]

Step 7: Selected features
     year   fips county.x   state.x  percent_food_insecure  \
189  2022  36000    Total  New York                     11
126  2023  36000    Total  New York                     10
63   2024  36000    Total  New York                     11
0    2025  36000    Total  New York                     13
190  2022  36001   Albany  New York                     10

     percent_household_income_required_for_child_care_expenses  \
189                                           36.26455
126                                           32.00000
63                                            38.00000
0                                             38.00000
190                                           36.26455

     food_environment_index  percent_fair_or_poor_health  percent_unemployed  \
189                     9.0                           16                10.0
126                     8.9                           12                 6.9
63                      8.6                           14                 4.3
0                       8.7                           16                 4.2
190                     8.3                           15                 7.2

     percent_children_in_poverty  …  percent_65_and_over  \
189                           17  …                 17.4
126                           19  …                 17.5
63                            19  …                 18.1
0                             19  …                 18.6
190                           13  …                 17.9

     percent_not_proficient_in_english  segregation_index  teen_birth_rate  \
189                                  7               0.35             13.0
126                                  7               0.34             13.0
63                                   7               0.34             11.0
0                                    7               0.33             10.0
190                                  2               0.21              9.0
```

```
     percent_children_in_single_parent_households  percent_low_birthweight  \
189                                      22.248677                 8.000000
126                                      22.248677                 8.000000
63                                       26.000000                 8.000000
0                                        26.000000                 7.292994
190                                      22.248677                 8.000000

     percent_black  rural_urban  food_insecure_lag1  food_insecure_lag2
189      14.400000            0                11.0                11.0
126      14.400000            0                11.0                11.0
63       14.400000            0                10.0                11.0
0         6.134286            0                11.0                10.0
190      12.900000            0                10.0                12.0

[5 rows x 33 columns]

Step 8: Data Analysis
Years available: [2022, 2023, 2024, 2025]
Counties with data: 63

Minimum years per county: 4
Maximum years per county: 4
Using n_steps = 1

Training years: [2022, 2023]
Test year: [2024]

Training data shape: (63, 1, 28)
Training target shape: (63,)

Test data shape: (63, 1, 28)
Test target shape: (63,)
Epoch 1/100
```

```
c:\Users\jashb\Lib\site-packages\keras\src\layers\rnn\rnn.py:200: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When using
Sequential models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(**kwargs)
```

```
2/2              3s 585ms/step - loss:
138.6572 - val_loss: 109.4728
Epoch 2/100
2/2              0s 54ms/step - loss:
143.1701 - val_loss: 109.2129
Epoch 3/100
2/2              0s 51ms/step - loss:
137.9595 - val_loss: 108.9411
Epoch 4/100
```

```
2/2            0s 51ms/step - loss:
142.8934 - val_loss: 108.6512
Epoch 5/100
2/2            0s 53ms/step - loss:
142.4121 - val_loss: 108.3426
Epoch 6/100
2/2            0s 58ms/step - loss:
141.4838 - val_loss: 108.0132
Epoch 7/100
2/2            0s 62ms/step - loss:
140.7425 - val_loss: 107.6594
Epoch 8/100
2/2            0s 58ms/step - loss:
137.3974 - val_loss: 107.2754
Epoch 9/100
2/2            0s 63ms/step - loss:
139.0445 - val_loss: 106.8514
Epoch 10/100
2/2            0s 62ms/step - loss:
135.0317 - val_loss: 106.3797
Epoch 11/100
2/2            0s 57ms/step - loss:
140.1931 - val_loss: 105.8503
Epoch 12/100
2/2            0s 57ms/step - loss:
138.9033 - val_loss: 105.2521
Epoch 13/100
2/2            0s 58ms/step - loss:
136.8620 - val_loss: 104.5711
Epoch 14/100
2/2            0s 55ms/step - loss:
132.2716 - val_loss: 103.7887
Epoch 15/100
2/2            0s 55ms/step - loss:
137.5652 - val_loss: 102.8917
Epoch 16/100
2/2            0s 57ms/step - loss:
135.7354 - val_loss: 101.8652
Epoch 17/100
2/2            0s 56ms/step - loss:
132.5397 - val_loss: 100.6818
Epoch 18/100
2/2            0s 53ms/step - loss:
132.8964 - val_loss: 99.3144
Epoch 19/100
2/2            0s 58ms/step - loss:
126.8001 - val_loss: 97.7308
Epoch 20/100
```

```
2/2             0s 55ms/step - loss:
124.6101 - val_loss: 95.8905
Epoch 21/100
2/2             0s 56ms/step - loss:
125.6228 - val_loss: 93.7459
Epoch 22/100
2/2             0s 56ms/step - loss:
121.0196 - val_loss: 91.2517
Epoch 23/100
2/2             0s 55ms/step - loss:
119.8000 - val_loss: 88.3413
Epoch 24/100
2/2             0s 58ms/step - loss:
113.4216 - val_loss: 84.9574
Epoch 25/100
2/2             0s 57ms/step - loss:
109.5356 - val_loss: 81.0235
Epoch 26/100
2/2             0s 56ms/step - loss:
106.3027 - val_loss: 76.4675
Epoch 27/100
2/2             0s 57ms/step - loss:
102.6778 - val_loss: 71.2334
Epoch 28/100
2/2             0s 54ms/step - loss:
94.9555 - val_loss: 65.2756
Epoch 29/100
2/2             0s 56ms/step - loss:
84.8758 - val_loss: 58.5604
Epoch 30/100
2/2             0s 65ms/step - loss:
74.4368 - val_loss: 51.1300
Epoch 31/100
2/2             0s 59ms/step - loss:
67.8760 - val_loss: 43.0760
Epoch 32/100
2/2             0s 53ms/step - loss:
53.9291 - val_loss: 34.6245
Epoch 33/100
2/2             0s 61ms/step - loss:
44.2650 - val_loss: 26.1006
Epoch 34/100
2/2             0s 66ms/step - loss:
36.3070 - val_loss: 18.0767
Epoch 35/100
2/2             0s 63ms/step - loss:
25.7063 - val_loss: 11.1809
Epoch 36/100
```

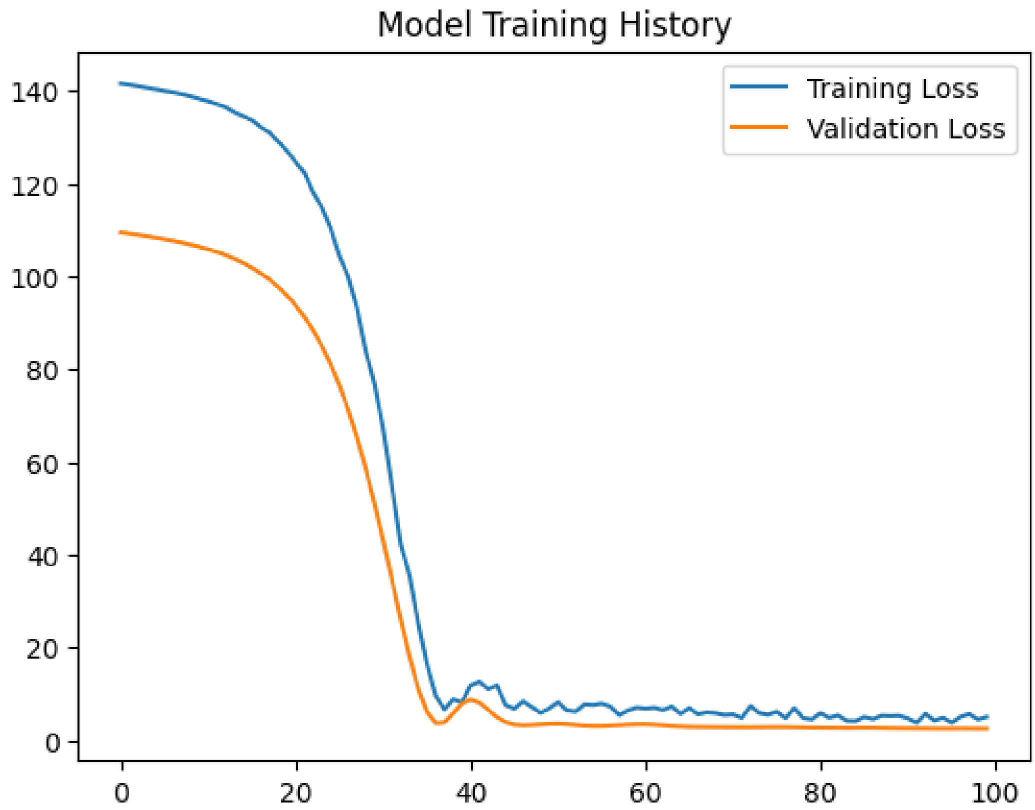
```
2/2              0s 56ms/step - loss:
16.7834 - val_loss: 6.2042
Epoch 37/100
2/2              0s 54ms/step - loss:
10.0800 - val_loss: 3.7206
Epoch 38/100
2/2              0s 52ms/step - loss:
6.7181 - val_loss: 3.8670
Epoch 39/100
2/2              0s 54ms/step - loss:
7.6601 - val_loss: 5.8433
Epoch 40/100
2/2              0s 55ms/step - loss:
7.6808 - val_loss: 7.9571
Epoch 41/100
2/2              0s 64ms/step - loss:
11.9945 - val_loss: 8.7960
Epoch 42/100
2/2              0s 58ms/step - loss:
12.9242 - val_loss: 8.1403
Epoch 43/100
2/2              0s 54ms/step - loss:
11.5004 - val_loss: 6.6087
Epoch 44/100
2/2              0s 53ms/step - loss:
12.2016 - val_loss: 5.0128
Epoch 45/100
2/2              0s 52ms/step - loss:
7.7544 - val_loss: 3.9272
Epoch 46/100
2/2              0s 51ms/step - loss:
7.0484 - val_loss: 3.4137
Epoch 47/100
2/2              0s 52ms/step - loss:
8.4630 - val_loss: 3.2639
Epoch 48/100
2/2              0s 51ms/step - loss:
7.1145 - val_loss: 3.3356
Epoch 49/100
2/2              0s 53ms/step - loss:
5.7704 - val_loss: 3.4704
Epoch 50/100
2/2              0s 51ms/step - loss:
7.1670 - val_loss: 3.5874
Epoch 51/100
2/2              0s 52ms/step - loss:
8.4340 - val_loss: 3.6347
Epoch 52/100
```

```
2/2              0s 53ms/step - loss:
6.4720 - val_loss: 3.5179
Epoch 53/100
2/2              0s 53ms/step - loss:
6.5044 - val_loss: 3.3377
Epoch 54/100
2/2              0s 54ms/step - loss:
7.7102 - val_loss: 3.1821
Epoch 55/100
2/2              0s 53ms/step - loss:
7.5351 - val_loss: 3.1139
Epoch 56/100
2/2              0s 49ms/step - loss:
8.3839 - val_loss: 3.1215
Epoch 57/100
2/2              0s 55ms/step - loss:
7.3458 - val_loss: 3.1986
Epoch 58/100
2/2              0s 50ms/step - loss:
5.1613 - val_loss: 3.2966
Epoch 59/100
2/2              0s 54ms/step - loss:
5.8214 - val_loss: 3.4163
Epoch 60/100
2/2              0s 53ms/step - loss:
7.1770 - val_loss: 3.4874
Epoch 61/100
2/2              0s 49ms/step - loss:
7.0529 - val_loss: 3.4960
Epoch 62/100
2/2              0s 58ms/step - loss:
7.6125 - val_loss: 3.4113
Epoch 63/100
2/2              0s 56ms/step - loss:
6.3878 - val_loss: 3.2746
Epoch 64/100
2/2              0s 56ms/step - loss:
7.4552 - val_loss: 3.1152
Epoch 65/100
2/2              0s 52ms/step - loss:
5.4546 - val_loss: 3.0254
Epoch 66/100
2/2              0s 55ms/step - loss:
7.2294 - val_loss: 2.9606
Epoch 67/100
2/2              0s 55ms/step - loss:
5.8877 - val_loss: 2.9205
Epoch 68/100
```

```
2/2              0s 53ms/step - loss:
5.6577 - val_loss: 2.9013
Epoch 69/100
2/2              0s 52ms/step - loss:
5.9458 - val_loss: 2.8878
Epoch 70/100
2/2              0s 54ms/step - loss:
5.7968 - val_loss: 2.8731
Epoch 71/100
2/2              0s 53ms/step - loss:
5.8743 - val_loss: 2.8591
Epoch 72/100
2/2              0s 49ms/step - loss:
4.7755 - val_loss: 2.8504
Epoch 73/100
2/2              0s 50ms/step - loss:
7.2101 - val_loss: 2.8538
Epoch 74/100
2/2              0s 56ms/step - loss:
5.9096 - val_loss: 2.8656
Epoch 75/100
2/2              0s 52ms/step - loss:
5.4682 - val_loss: 2.8807
Epoch 76/100
2/2              0s 50ms/step - loss:
5.7183 - val_loss: 2.8951
Epoch 77/100
2/2              0s 52ms/step - loss:
4.8998 - val_loss: 2.8773
Epoch 78/100
2/2              0s 53ms/step - loss:
6.9327 - val_loss: 2.8455
Epoch 79/100
2/2              0s 55ms/step - loss:
4.7884 - val_loss: 2.8114
Epoch 80/100
2/2              0s 53ms/step - loss:
4.6939 - val_loss: 2.7804
Epoch 81/100
2/2              0s 56ms/step - loss:
5.3595 - val_loss: 2.7540
Epoch 82/100
2/2              0s 66ms/step - loss:
4.6416 - val_loss: 2.7334
Epoch 83/100
2/2              0s 54ms/step - loss:
5.2723 - val_loss: 2.7201
Epoch 84/100
```

```
2/2            0s 53ms/step - loss:
4.0609 - val_loss: 2.7148
Epoch 85/100
2/2            0s 49ms/step - loss:
3.9861 - val_loss: 2.7207
Epoch 86/100
2/2            0s 55ms/step - loss:
5.2969 - val_loss: 2.7320
Epoch 87/100
2/2            0s 49ms/step - loss:
4.5736 - val_loss: 2.7297
Epoch 88/100
2/2            0s 55ms/step - loss:
5.4222 - val_loss: 2.7031
Epoch 89/100
2/2            0s 54ms/step - loss:
5.3620 - val_loss: 2.6739
Epoch 90/100
2/2            0s 57ms/step - loss:
5.4869 - val_loss: 2.6529
Epoch 91/100
2/2            0s 64ms/step - loss:
4.6129 - val_loss: 2.6425
Epoch 92/100
2/2            0s 60ms/step - loss:
3.8069 - val_loss: 2.6298
Epoch 93/100
2/2            0s 53ms/step - loss:
5.3095 - val_loss: 2.6144
Epoch 94/100
2/2            0s 51ms/step - loss:
4.1286 - val_loss: 2.6046
Epoch 95/100
2/2            0s 52ms/step - loss:
4.6150 - val_loss: 2.5942
Epoch 96/100
2/2            0s 51ms/step - loss:
3.6085 - val_loss: 2.5953
Epoch 97/100
2/2            0s 51ms/step - loss:
4.7717 - val_loss: 2.6055
Epoch 98/100
2/2            0s 56ms/step - loss:
5.6333 - val_loss: 2.5943
Epoch 99/100
2/2            0s 50ms/step - loss:
4.1112 - val_loss: 2.5828
Epoch 100/100
```

```
2/2              0s 53ms/step - loss:
5.0832 - val_loss: 2.5628
2/2              0s 185ms/step
2/2              0s 13ms/step

Train RMSE: 1.2394944352508956
Test RMSE: 2.1655362983327997
```



5.1 LSTM 4 : Metrics Table

[ ]:
```
import pandas as pd

# Calculate MSE
mse_train = mean_squared_error(y_train, train_pred)
mse_test = mean_squared_error(y_test, test_pred)

rmse_train = np.sqrt(mse_train)
rmse_test = np.sqrt(mse_test)

# Calculate MAPE
mape_train = np.mean(np.abs((y_train - train_pred.flatten()) / y_train)) * 100
```

```
mape_test = np.mean(np.abs((y_test - test_pred.flatten()) / y_test)) * 100

# Create a table
results = pd.DataFrame({
    "Metric": ["MSE", "RMSE", "MAPE (%)"],
    "Train": [mse_train, rmse_train, mape_train],
    "Test": [mse_test, rmse_test, mape_test]
})

print("LSTM 4 Model Metrics: Early Stopping\n")
print(results)
```

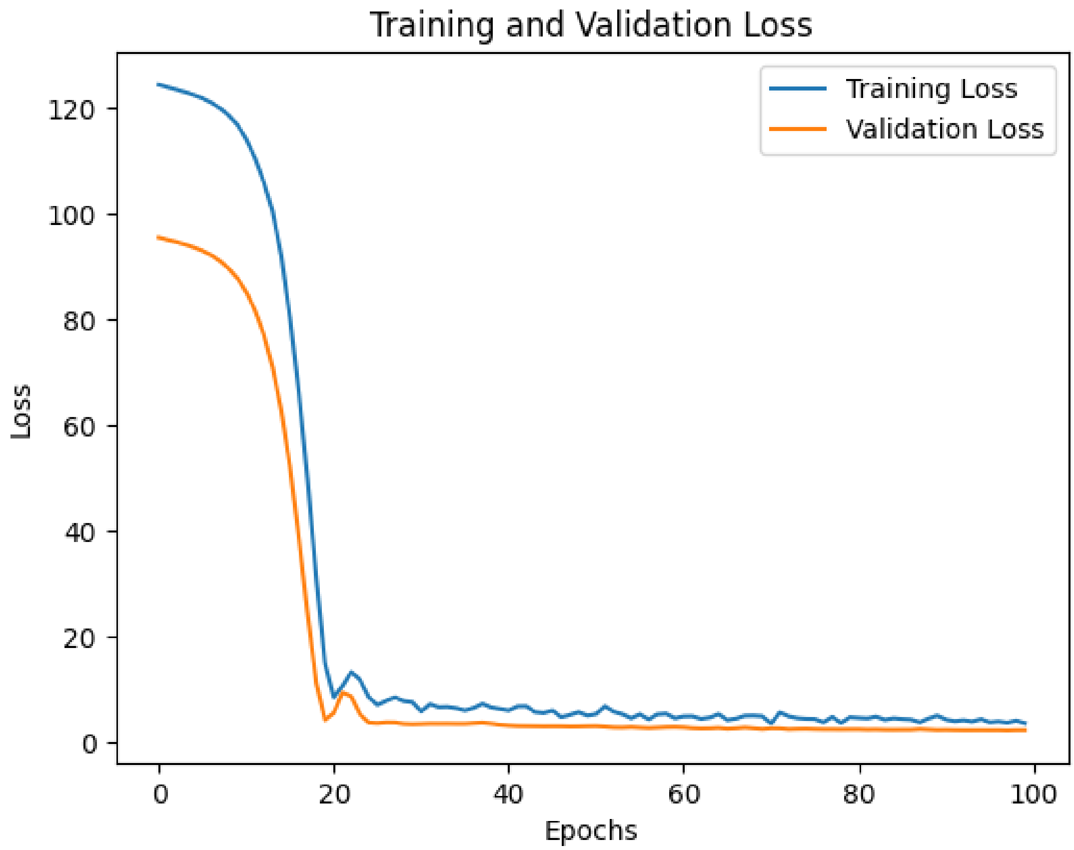
```
LSTM 4 Model Metrics: Early Stopping

     Metric     Train       Test
0       MSE  1.098105   5.054720
1      RMSE  1.047905   2.248270
2  MAPE (%)  8.393658  16.058777
```

5.2 Visualizing the best performing LSTM Model

- This model was an L2 regularized model, with early stopping added to aid the model from overfitting.

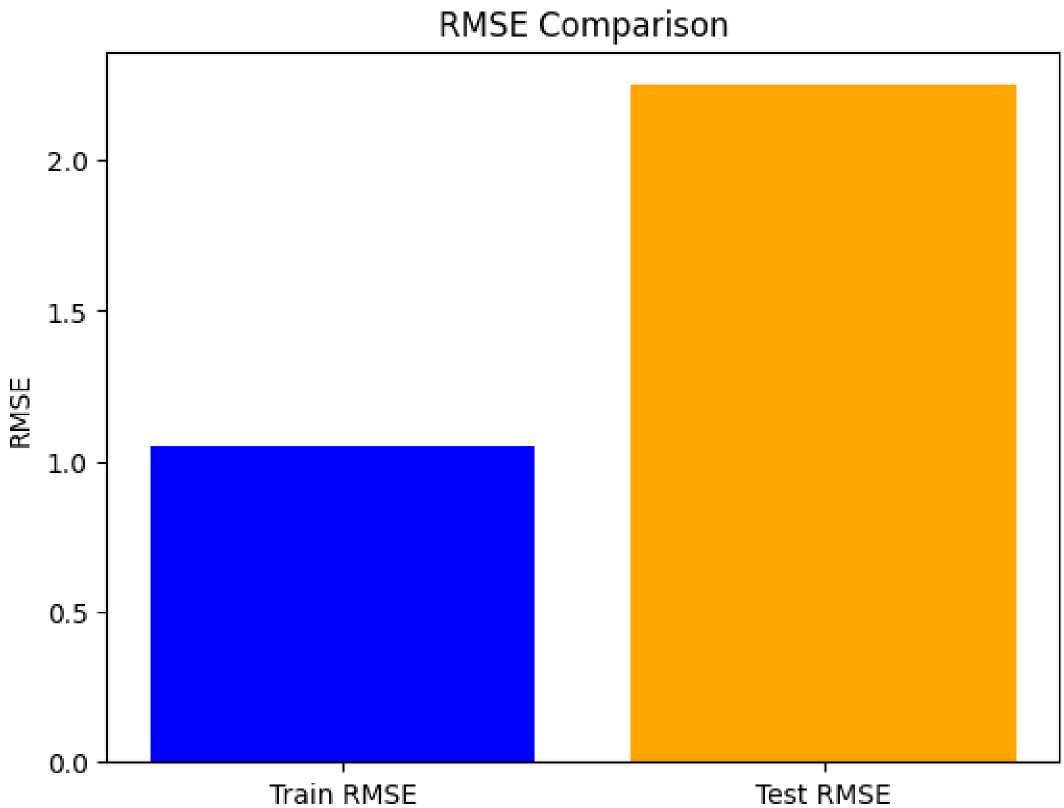
1. Training and Validation Loss Curve

[ ]:
```
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.legend()
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.show()
```



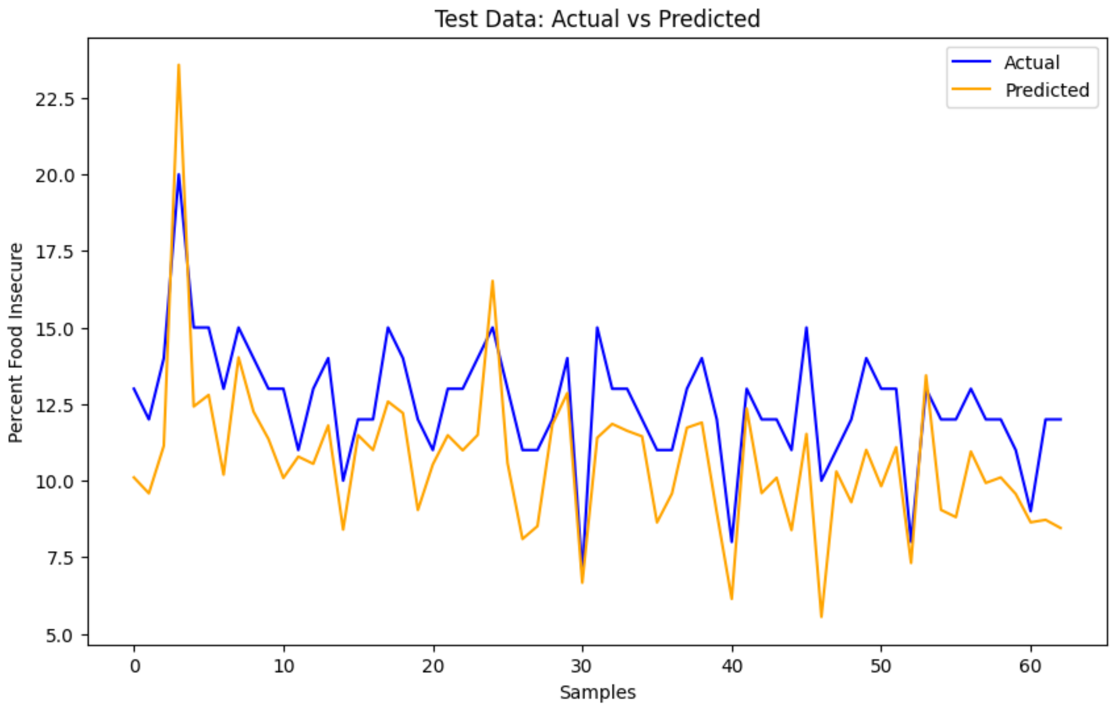
2. RMSE Comparison (Train vs Test)

[ ]:
```
train_rmse = np.sqrt(mean_squared_error(y_train, train_pred))
test_rmse = np.sqrt(mean_squared_error(y_test, test_pred))

plt.bar(['Train RMSE', 'Test RMSE'], [train_rmse, test_rmse], color=['blue',␣
  ↪'orange'])
plt.title('RMSE Comparison')
plt.ylabel('RMSE')
plt.show()
```



3. Predicted vs Actual (Test Data)

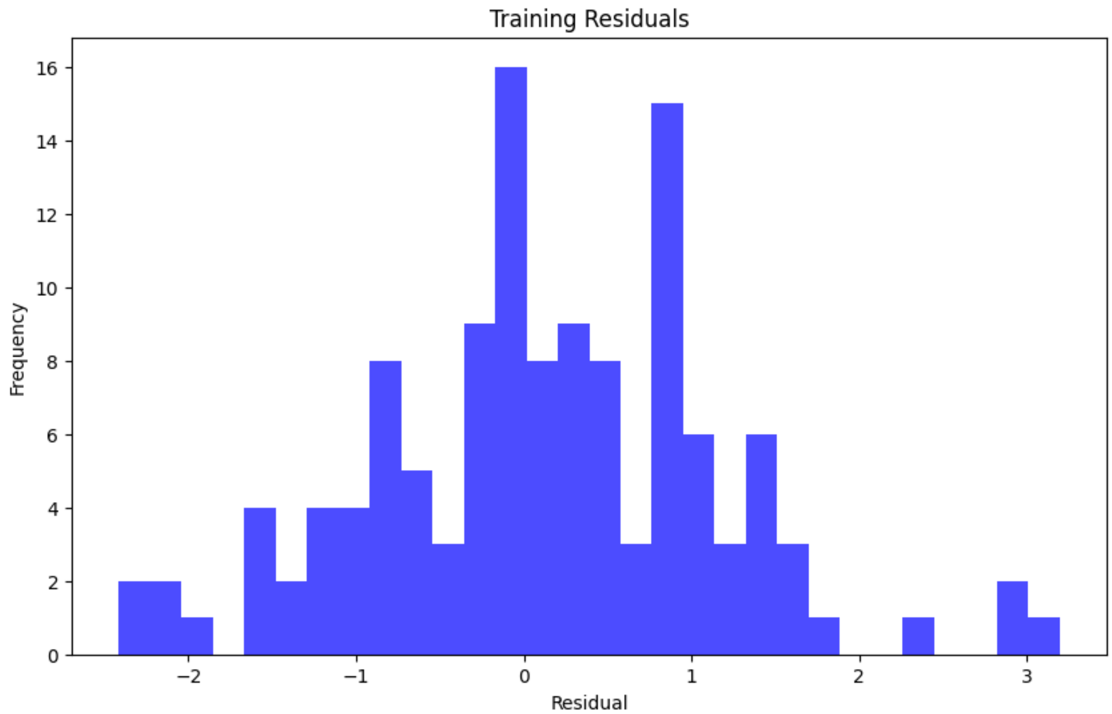
[ ]:
```
plt.figure(figsize=(10, 6))
plt.plot(y_test, label='Actual', color='blue')
plt.plot(test_pred, label='Predicted', color='orange')
plt.title('Test Data: Actual vs Predicted')
plt.xlabel('Samples')
plt.ylabel('Percent Food Insecure')
plt.legend()
plt.show()
```



4. Residual Analysis (Train and Test)

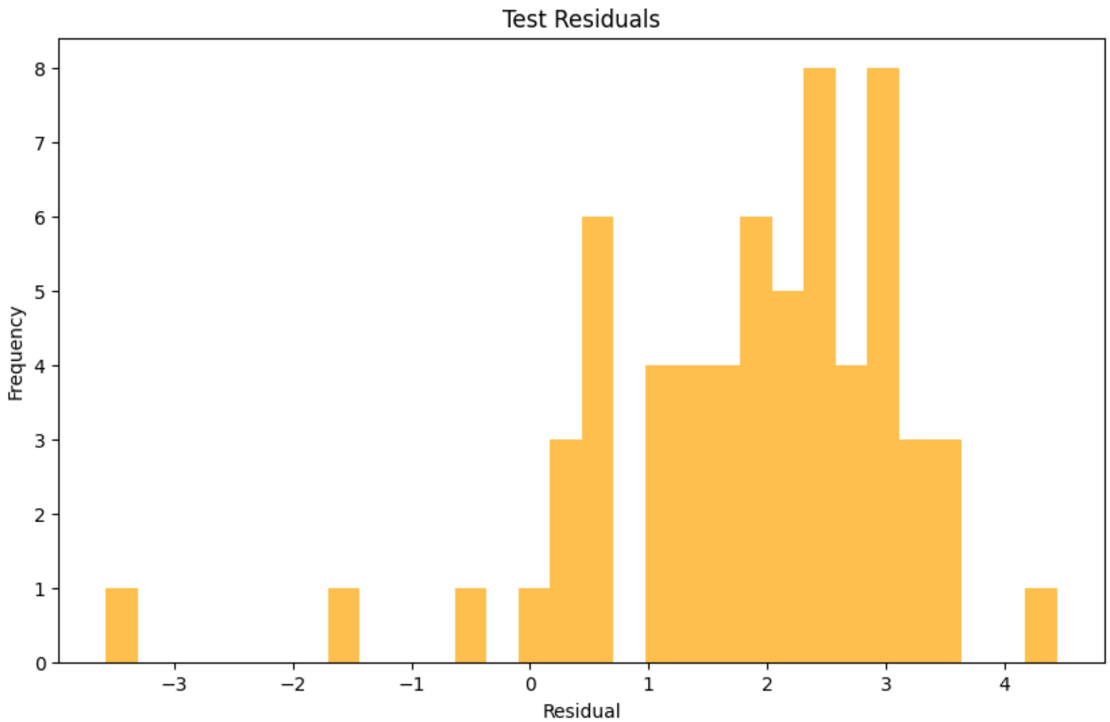
Training Resid

[ ]:
```
train_residuals = y_train - train_pred.flatten()
plt.figure(figsize=(10, 6))
plt.hist(train_residuals, bins=30, color='blue', alpha=0.7)
plt.title('Training Residuals')
plt.xlabel('Residual')
plt.ylabel('Frequency')
plt.show()
```



Testing Resid

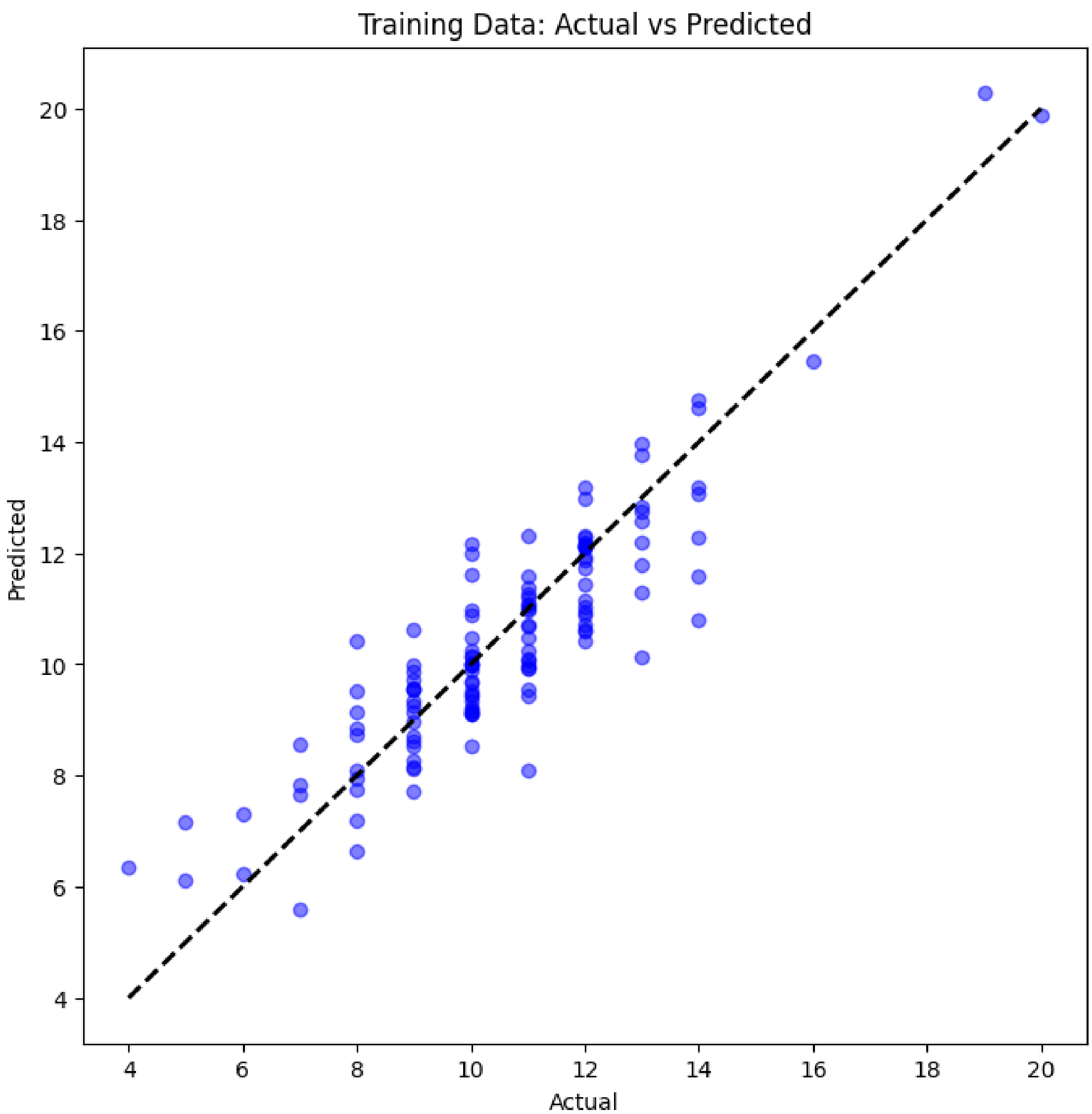
```
[ ]: test_residuals = y_test - test_pred.flatten()
     plt.figure(figsize=(10, 6))
     plt.hist(test_residuals, bins=30, color='orange', alpha=0.7)
     plt.title('Test Residuals')
     plt.xlabel('Residual')
     plt.ylabel('Frequency')
     plt.show()
```



5. Scatter Plot: Actual vs Predicted

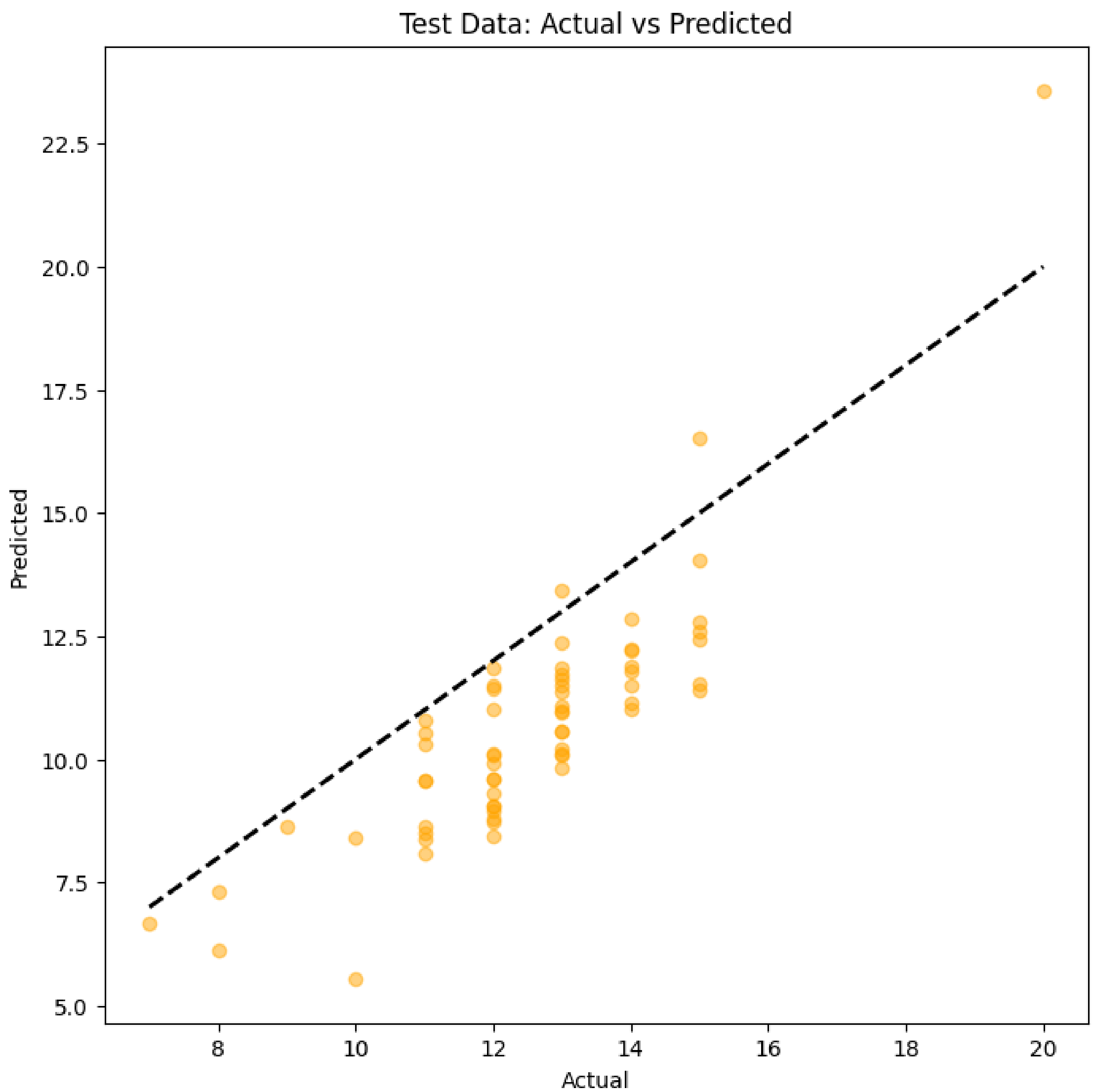
Training Data

[ ]:
```
plt.figure(figsize=(8, 8))
plt.scatter(y_train, train_pred, alpha=0.5, color='blue')
plt.plot([y_train.min(), y_train.max()], [y_train.min(), y_train.max()], 'k--',␣
 ↪lw=2)
plt.title('Training Data: Actual vs Predicted')
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.show()
```



Test Data

[ ]:
```
plt.figure(figsize=(8, 8))
plt.scatter(y_test, test_pred, alpha=0.5, color='orange')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--',␣
↪lw=2)
plt.title('Test Data: Actual vs Predicted')
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.show()
```



6. Predicted Values Table

```
[ ]: predicted_vs_actual = pd.DataFrame({
         'Actual': y_test,
         'Predicted': test_pred.flatten()
     })
     print(predicted_vs_actual).head()
```

```
   Actual  Predicted
0      13  10.104425
1      12   9.586191
2      14  11.137208
3      20  23.577702
4      15  12.421782
```

```
..       ...        ...
58       12   10.105597
59       11    9.566097
60        9    8.639240
61       12    8.720520
62       12    8.454118

[63 rows x 2 columns]
```

7. Top Counties with Highest Predicted Food Insecurity

[ ]:
```
test['Predicted'] = test_pred.flatten()
top_10_counties = test.nlargest(10, 'Predicted')

plt.figure(figsize=(12, 6))
plt.bar(top_10_counties['county.x'], top_10_counties['Predicted'],␣
  ↪color='firebrick')
plt.title('Top 10 Counties with Highest Predicted Food Insecurity')
plt.xlabel('County')
plt.ylabel('Predicted Food Insecurity (%)')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
C:\Users\jashb\AppData\Local\Temp\ipykernel_1536\1192875939.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  test['Predicted'] = test_pred.flatten()
```