In [27]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.multioutput import MultiOutputRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# Load the Food Sold dataset
raw = pd.read_csv('Food Sold everyday.csv')
raw.head
```

Out[27]:
```
<bound method NDFrame.head of           Date Day of the week  Day  Month  Year  T
highs  Tenders  Wings  \
0      3/1/2017       Wednesday    1      3  2017       8       11      7
1      3/2/2017        Thursday    2      3  2017      11       27     13
2      3/3/2017          Friday    3      3  2017       9       26     12
3      3/4/2017        Saturday    4      3  2017       9       17     11
4      3/5/2017          Sunday    5      3  2017       7        3      8
...         ...             ...  ...    ...   ...     ...      ...    ...
2725  8/16/2024          Friday   16      8  2024       7       41     14
2726  8/17/2024        Saturday   17      8  2024       5       28      8
2727  8/18/2024          Sunday   18      8  2024       5       10      4
2728  8/19/2024          Monday   19      8  2024       6       24     11
2729  8/20/2024         Tuesday   20      8  2024       6        1      9

      Egg rolls  Crispitos  ...  Quesadilla  Is Holiday  Chicken  FriedSnacks  \
0             1         21  ...           8       False       26           39
1             5         12  ...           1       False       51           45
2             6         13  ...           9       False       47           44
3             4         19  ...           8       False       37           25
4             1         15  ...           2       False       18           18
...         ...        ...  ...         ...         ...      ...          ...
2725          5         21  ...           5       False       62           48
2726          1          5  ...           6       False       41           23
2727          4         16  ...           4       False       19           21
2728          3         19  ...           3       False       41           36
2729          3         16  ...           1       False       16           45

      FriedBurritos  Otherfood  Price_Chicken  Price_FriedSnacks  \
0                65         31           4.99               1.79
1                63         40           4.99               1.79
2                47         65           4.99               1.79
3                65         47           4.99               1.79
4                43         27           4.99               1.79
...             ...        ...            ...                ...
2725             60         52           7.99               2.48
2726             38         36           7.99               2.48
2727             39         24           7.99               2.48
2728             56         26           7.99               2.48
2729             59         35           7.99               2.48

      Price_FriedBurritos  Price_Other_Food
0                    2.29              2.48
1                    2.29              2.48
2                    2.29              2.48
3                    2.29              2.48
4                    2.29              2.48
...                   ...               ...
2725                 2.99              3.49
2726                 2.99              3.49
2727                 2.99              3.49
2728                 2.99              3.49
2729                 2.99              3.49

[2730 rows x 32 columns]>
```

In [28]:
```
columns = ['Date','Chicken','FriedSnacks','FriedBurritos','Otherfood','Is Holiday',
           'Price_FriedSnacks','Price_FriedBurritos','Price_Other_Food']
df = raw[columns]
df
```

Out[28]:

	Date	Chicken	FriedSnacks	FriedBurritos	Otherfood	Is Holiday	Price_Chicken	F
0	3/1/2017	26	39	65	31	False	4.99	
1	3/2/2017	51	45	63	40	False	4.99	
2	3/3/2017	47	44	47	65	False	4.99	
3	3/4/2017	37	25	65	47	False	4.99	
4	3/5/2017	18	18	43	27	False	4.99	
...	...	...	...	...	...	...	...	
2725	8/16/2024	62	48	60	52	False	7.99	
2726	8/17/2024	41	23	38	36	False	7.99	
2727	8/18/2024	19	21	39	24	False	7.99	
2728	8/19/2024	41	36	56	26	False	7.99	
2729	8/20/2024	16	45	59	35	False	7.99	

2730 rows × 10 columns

In [29]:
```
# 1. Convert the 'Date' column to datetime
df['Date'] = pd.to_datetime(df['Date'])
```

```
C:\Users\harsh\AppData\Local\Temp\ipykernel_20232\3330707552.py:2: SettingWithCopyWa
rning:
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/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  df['Date'] = pd.to_datetime(df['Date'])
```

In [30]:
```
# 2. Set the date as the index (optional but helpful for time series)
df.set_index('Date', inplace=True)
```

In [31]:
```
df
```

Out[31]:

Date	Chicken	FriedSnacks	FriedBurritos	Otherfood	Is Holiday	Price_Chicken	Price_FriedS
2017-03-01	26	39	65	31	False	4.99	
2017-03-02	51	45	63	40	False	4.99	
2017-03-03	47	44	47	65	False	4.99	
2017-03-04	37	25	65	47	False	4.99	
2017-03-05	18	18	43	27	False	4.99	
...	...	...	...	...	...	...	
2024-08-16	62	48	60	52	False	7.99	
2024-08-17	41	23	38	36	False	7.99	
2024-08-18	19	21	39	24	False	7.99	
2024-08-19	41	36	56	26	False	7.99	
2024-08-20	16	45	59	35	False	7.99	

2730 rows × 9 columns

In [32]:
```
# 3. Handle missing data (for simplicity, filling missing values with median)
df.fillna(df.median(), inplace=True)
```

```
C:\Users\harsh\AppData\Local\Temp\ipykernel_20232\3090981337.py:2: SettingWithCopyWa
rning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  df.fillna(df.median(), inplace=True)
```

In [33]:
```
# 4. Create additional time-related features
df['Day_of_Week'] = df.index.dayofweek  # Monday = 0, Sunday = 6
df['Month'] = df.index.month  # Month as an integer
df['Weekday'] = np.where(df['Day_of_Week'] < 5, 1, 0)  # Weekday vs Weekend
```

```
df['Is_Holiday'] = df['Is Holiday']  # Assuming you already have a holiday flag
df['Is_Weekend'] = np.where(df['Day_of_Week'] >= 5, 1, 0)  # Weekend indicator
```

```
C:\Users\harsh\AppData\Local\Temp\ipykernel_20232\969340903.py:2: SettingWithCopyWar
ning:
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/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  df['Day_of_Week'] = df.index.dayofweek  # Monday = 0, Sunday = 6
C:\Users\harsh\AppData\Local\Temp\ipykernel_20232\969340903.py:3: SettingWithCopyWar
ning:
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/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  df['Month'] = df.index.month  # Month as an integer
C:\Users\harsh\AppData\Local\Temp\ipykernel_20232\969340903.py:4: SettingWithCopyWar
ning:
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/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  df['Weekday'] = np.where(df['Day_of_Week'] < 5, 1, 0)  # Weekday vs Weekend
C:\Users\harsh\AppData\Local\Temp\ipykernel_20232\969340903.py:5: SettingWithCopyWar
ning:
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/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  df['Is_Holiday'] = df['Is Holiday']  # Assuming you already have a holiday flag
C:\Users\harsh\AppData\Local\Temp\ipykernel_20232\969340903.py:6: SettingWithCopyWar
ning:
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/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  df['Is_Weekend'] = np.where(df['Day_of_Week'] >= 5, 1, 0)  # Weekend indicator
```

In [34]:
```
# 5. Lag features (Previous week's sales)
df['Lag_Chicken_Wings'] = df['Chicken'].shift(7)  # 7-day lag
df['Lag_Breakfast'] = df['FriedSnacks'].shift(7)  # 7-day lag
df['Lag_Fried_Burritos'] = df['FriedBurritos'].shift(7)  # 7-day lag
df['Lag_Other_Food'] = df['Otherfood'].shift(7)  # 7-day lag
```

```
C:\Users\harsh\AppData\Local\Temp\ipykernel_20232\2124679577.py:2: SettingWithCopyWa
rning:
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/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  df['Lag_Chicken_Wings'] = df['Chicken'].shift(7)  # 7-day lag
C:\Users\harsh\AppData\Local\Temp\ipykernel_20232\2124679577.py:3: SettingWithCopyWa
rning:
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/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  df['Lag_Breakfast'] = df['FriedSnacks'].shift(7)  # 7-day lag
C:\Users\harsh\AppData\Local\Temp\ipykernel_20232\2124679577.py:4: SettingWithCopyWa
rning:
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/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  df['Lag_Fried_Burritos'] = df['FriedBurritos'].shift(7)  # 7-day lag
C:\Users\harsh\AppData\Local\Temp\ipykernel_20232\2124679577.py:5: SettingWithCopyWa
rning:
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/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  df['Lag_Other_Food'] = df['Otherfood'].shift(7)  # 7-day lag
```

In [35]:
```
df
```

Out[35]:

Date	Chicken	FriedSnacks	FriedBurritos	Otherfood	Is Holiday	Price_Chicken	Price_FriedS
2017-03-01	26	39	65	31	False	4.99	
2017-03-02	51	45	63	40	False	4.99	
2017-03-03	47	44	47	65	False	4.99	
2017-03-04	37	25	65	47	False	4.99	
2017-03-05	18	18	43	27	False	4.99	
...	...	...	...	...	...	...	
2024-08-16	62	48	60	52	False	7.99	
2024-08-17	41	23	38	36	False	7.99	
2024-08-18	19	21	39	24	False	7.99	
2024-08-19	41	36	56	26	False	7.99	
2024-08-20	16	45	59	35	False	7.99	

2730 rows × 18 columns

In [36]:
```
# 6. Moving average features (for smoothing the demand pattern)
df['MA_Chicken_Wings'] = df['Chicken'].rolling(window=7).mean()  # 7-day moving ave
df['MA_Breakfast'] = df['FriedSnacks'].rolling(window=7).mean()
df['MA_Fried_Burritos'] = df['FriedBurritos'].rolling(window=7).mean()
df['MA_Other_Food'] = df['Otherfood'].rolling(window=7).mean()
```

```
C:\Users\harsh\AppData\Local\Temp\ipykernel_20232\2299311345.py:2: SettingWithCopyWa
rning:
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/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  df['MA_Chicken_Wings'] = df['Chicken'].rolling(window=7).mean()  # 7-day moving av
erage
C:\Users\harsh\AppData\Local\Temp\ipykernel_20232\2299311345.py:3: SettingWithCopyWa
rning:
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/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  df['MA_Breakfast'] = df['FriedSnacks'].rolling(window=7).mean()
C:\Users\harsh\AppData\Local\Temp\ipykernel_20232\2299311345.py:4: SettingWithCopyWa
rning:
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/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  df['MA_Fried_Burritos'] = df['FriedBurritos'].rolling(window=7).mean()
C:\Users\harsh\AppData\Local\Temp\ipykernel_20232\2299311345.py:5: SettingWithCopyWa
rning:
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/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  df['MA_Other_Food'] = df['Otherfood'].rolling(window=7).mean()
```

In [37]:
```
# 7. Drop rows with NaN values (due to shifting and rolling)
df.dropna(inplace=True)
```

```
C:\Users\harsh\AppData\Local\Temp\ipykernel_20232\3566349350.py:2: SettingWithCopyWa
rning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  df.dropna(inplace=True)
```

In [38]:
```
df
```

Out[38]:

Date	Chicken	FriedSnacks	FriedBurritos	Otherfood	Is Holiday	Price_Chicken	Price_FriedS
2017-03-08	28	30	55	43	False	4.99	
2017-03-09	51	42	68	43	False	4.99	
2017-03-10	54	41	91	46	False	4.99	
2017-03-11	47	38	72	28	False	4.99	
2017-03-12	23	31	32	26	False	4.99	
...	...	...	...	...	...	...	
2024-08-16	62	48	60	52	False	7.99	
2024-08-17	41	23	38	36	False	7.99	
2024-08-18	19	21	39	24	False	7.99	
2024-08-19	41	36	56	26	False	7.99	
2024-08-20	16	45	59	35	False	7.99	

2723 rows × 22 columns

In [39]:
```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

# 8. Scale numerical features (for models like XGBoost or LSTM)
scaler = StandardScaler()
```

In [40]:
```
df.columns
```

Out[40]:
```
Index(['Chicken', 'FriedSnacks', 'FriedBurritos', 'Otherfood', 'Is Holiday',
       'Price_Chicken', 'Price_FriedSnacks', 'Price_FriedBurritos',
       'Price_Other_Food', 'Day_of_Week', 'Month', 'Weekday', 'Is_Holiday',
       'Is_Weekend', 'Lag_Chicken_Wings', 'Lag_Breakfast',
       'Lag_Fried_Burritos', 'Lag_Other_Food', 'MA_Chicken_Wings',
       'MA_Breakfast', 'MA_Fried_Burritos', 'MA_Other_Food'],
      dtype='object')
```

In [41]:
```
# Features to scale (you can adjust based on your model needs)
columns_to_scale = ['Price_Chicken', 'Price_FriedSnacks',
                    'Price_FriedBurritos', 'Price_Other_Food', 'Lag_Chicken_Wings',
                    'Lag_Breakfast', 'Lag_Fried_Burritos', 'Lag_Other_Food', 'MA_Ch
                    'MA_Breakfast', 'MA_Fried_Burritos', 'MA_Other_Food']
```

In [42]:
```
df[columns_to_scale] = scaler.fit_transform(df[columns_to_scale])
```

```
C:\Users\harsh\AppData\Local\Temp\ipykernel_20232\2566292694.py:1: SettingWithCopyWa
rning:
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/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  df[columns_to_scale] = scaler.fit_transform(df[columns_to_scale])
```

In [43]:
```
# 9. Split the data into training and test sets
X = df.drop(['Chicken', 'FriedSnacks', 'FriedBurritos', 'Otherfood'], axis=1)
y = df[['Chicken', 'FriedSnacks', 'FriedBurritos', 'Otherfood']]  # Target variable
```

In [44]:
```
# Train-test split (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=Fa
```

In [45]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.multioutput import MultiOutputRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error
import xgboost as xgb


# Multi-output XGBoost
xgb_model = MultiOutputRegressor(xgb.XGBRegressor(
    objective='reg:squarederror',
    n_estimators=100,
    learning_rate=0.1,
    max_depth=6
))

xgb_model.fit(X_train, y_train)

# Predictions
y_pred_xgb = xgb_model.predict(X_test)

# Evaluation
mse_xgb = mean_squared_error(y_test, y_pred_xgb, multioutput='raw_values')
```

```
mae_xgb = mean_absolute_error(y_test, y_pred_xgb)

rmse_xgb = np.sqrt(mse_xgb)

print(f"XGBoost RMSE: {rmse_xgb}, MAE: {mae_xgb}")
```

```
XGBoost RMSE: [11.90588538  5.89057083  9.43858935  7.22221446], MAE: 6.924496826119
3796
```

In [54]:
```
y_test
```

Out[54]:

Date	Chicken	FriedSnacks	FriedBurritos	Otherfood
2023-05-26	37	37	43	55
2023-05-27	13	29	53	37
2023-05-28	25	17	36	22
2023-05-29	35	32	62	47
2023-05-30	59	37	51	37
...	...	...	...	...
2024-08-16	62	48	60	52
2024-08-17	41	23	38	36
2024-08-18	19	21	39	24
2024-08-19	41	36	56	26
2024-08-20	16	45	59	35

453 rows × 4 columns

In [55]:
```
y_pred_xgb
```

Out[55]:
```
array([[49.880768, 29.062195, 60.324074, 38.02705 ],
       [53.515903, 33.023   , 65.84573 , 43.223034],
       [34.882748, 29.304873, 56.0003  , 38.932697],
       ...,
       [43.18757 , 28.191128, 60.18996 , 36.915016],
       [25.04976 , 20.996548, 37.997463, 27.17291 ],
       [43.77469 , 25.596685, 53.14465 , 34.12285 ]], dtype=float32)
```

In [56]:
```
y_pred_xgb.shape
```

Out[56]:
```
(544, 4)
```

In [46]:
```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from sklearn.preprocessing import MinMaxScaler
```

```
# Scale features and targets (for LSTM, scaling is necessary)
scaler_X = MinMaxScaler()
scaler_y = MinMaxScaler()

X_scaled = scaler_X.fit_transform(X)
y_scaled = scaler_y.fit_transform(y)

# Create sequences for LSTM (assuming sequence_length = 7)
def create_sequences(data_X, data_y, sequence_length):
    X_seq, y_seq = [], []
    for i in range(len(data_X) - sequence_length):
        X_seq.append(data_X[i:i + sequence_length])
        y_seq.append(data_y[i + sequence_length])
    return np.array(X_seq), np.array(y_seq)

sequence_length = 7
X_seq, y_seq = create_sequences(X_scaled, y_scaled, sequence_length)

# Train-test split
split_index = int(X_seq.shape[0] * 0.8)
X_train, X_test = X_seq[:split_index], X_seq[split_index:]
y_train, y_test = y_seq[:split_index], y_seq[split_index:]

# Build LSTM Model
lstm_model = Sequential([
    LSTM(50, activation='relu', input_shape=(sequence_length, X_train.shape[2])),
    Dense(4)  # Output layer for 4 targets (multi-output)
])

lstm_model.compile(optimizer='adam', loss='mse')
lstm_model.fit(X_train, y_train, epochs=20, batch_size=16, validation_data=(X_test,

# Predictions
y_pred_lstm = lstm_model.predict(X_test)
y_pred_lstm = scaler_y.inverse_transform(y_pred_lstm)

# Evaluation
mse_lstm = mean_squared_error(scaler_y.inverse_transform(y_test), y_pred_lstm, mult
mae_lstm = mean_absolute_error(scaler_y.inverse_transform(y_test), y_pred_lstm)
rmse_lstm = np.sqrt(mse_lstm)

print(f"LSTM RMSE: {rmse_lstm}, MAE: {mae_lstm}")
```

```
Epoch 1/20
```

```
C:\Users\harsh\anaconda3\Lib\site-packages\keras\src\layers\rnn\rnn.py:200: UserWarn
ing: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequen
tial models, prefer using an `Input(shape)` object as the first layer in the model i
nstead.
  super().__init__(**kwargs)
```

```
136/136 ━━━━━━━━━━━━━━━━━━━━ 3s 6ms/step - loss: 0.0668 - val_loss: 0.0172
Epoch 2/20
136/136 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.0179 - val_loss: 0.0173
Epoch 3/20
136/136 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.0174 - val_loss: 0.0185
Epoch 4/20
136/136 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.0169 - val_loss: 0.0159
Epoch 5/20
136/136 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.0166 - val_loss: 0.0155
Epoch 6/20
136/136 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.0156 - val_loss: 0.0148
Epoch 7/20
136/136 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.0154 - val_loss: 0.0138
Epoch 8/20
136/136 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.0151 - val_loss: 0.0139
Epoch 9/20
136/136 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.0140 - val_loss: 0.0133
Epoch 10/20
136/136 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.0135 - val_loss: 0.0134
Epoch 11/20
136/136 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.0136 - val_loss: 0.0133
Epoch 12/20
136/136 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0126 - val_loss: 0.0140
Epoch 13/20
136/136 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0132 - val_loss: 0.0133
Epoch 14/20
136/136 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0134 - val_loss: 0.0133
Epoch 15/20
136/136 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - loss: 0.0135 - val_loss: 0.0134
Epoch 16/20
136/136 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - loss: 0.0131 - val_loss: 0.0136
Epoch 17/20
136/136 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - loss: 0.0130 - val_loss: 0.0134
Epoch 18/20
136/136 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0127 - val_loss: 0.0135
Epoch 19/20
136/136 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0132 - val_loss: 0.0130
Epoch 20/20
136/136 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0126 - val_loss: 0.0136
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
LSTM RMSE: [12.3740534   6.82773713 10.47337949  7.968758  ], MAE: 7.572232513743288
```

In [57]:
```
y_pred_lstm
```

Out[57]:
```
array([[44.290054, 40.622616, 69.972984, 42.343548],
       [37.140255, 33.115105, 57.296883, 35.239563],
       [27.20823 , 23.149181, 40.25482 , 25.886114],
       ...,
       [26.931171, 25.351841, 41.385143, 26.70866 ],
       [38.375603, 34.91643 , 58.977848, 37.033215],
       [37.893948, 34.46697 , 59.840946, 36.709084]], dtype=float32)
```

In [58]:
```
y_test
```

Out[58]:

Date	Chicken	FriedSnacks	FriedBurritos	Otherfood
2023-05-26	37	37	43	55
2023-05-27	13	29	53	37
2023-05-28	25	17	36	22
2023-05-29	35	32	62	47
2023-05-30	59	37	51	37
...	...	...	...	...
2024-08-16	62	48	60	52
2024-08-17	41	23	38	36
2024-08-18	19	21	39	24
2024-08-19	41	36	56	26
2024-08-20	16	45	59	35

453 rows × 4 columns

In [47]:
```
print("Model Comparison:")
print(f"XGBoost RMSE (per target): {rmse_xgb}")
print(f"LSTM RMSE (per target): {rmse_lstm}")

# Best model based on average RMSE across targets
avg_rmse_xgb = np.mean(rmse_xgb)
avg_rmse_lstm = np.mean(rmse_lstm)

if avg_rmse_xgb < avg_rmse_lstm:
    print("XGBoost is the best model.")
else:
    print("LSTM is the best model.")
```

```
Model Comparison:
XGBoost RMSE (per target): [11.90588538  5.89057083  9.43858935  7.22221446]
LSTM RMSE (per target): [12.3740534   6.82773713 10.47337949  7.968758  ]
XGBoost is the best model.
```

In [ ]:

In [ ]:

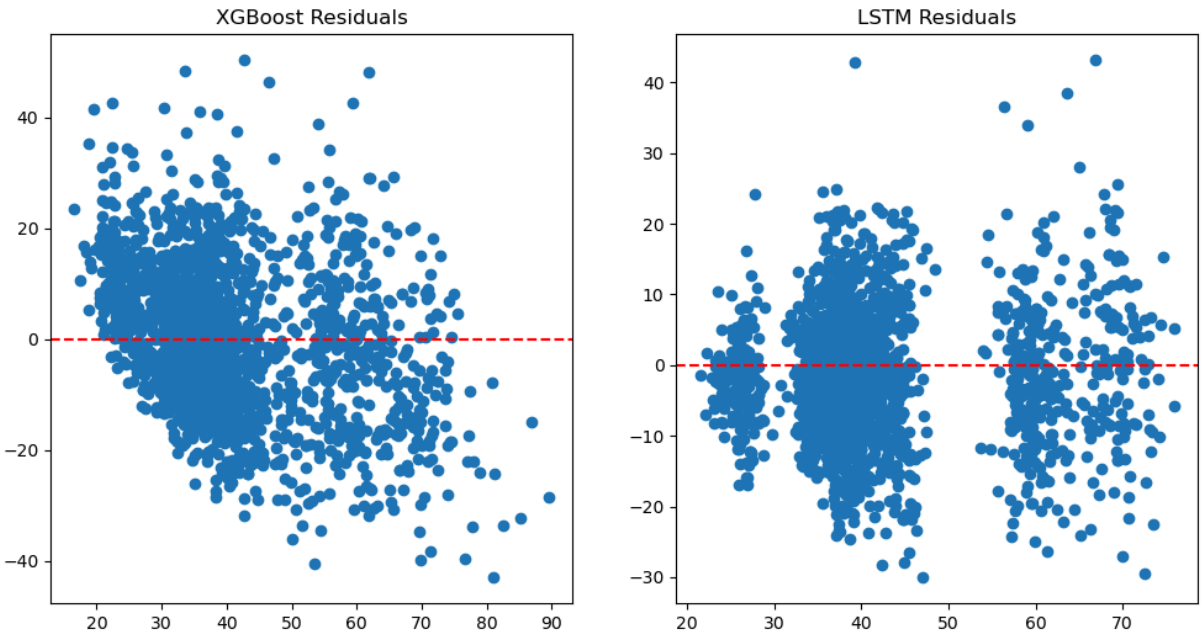
In [59]:
```
# Slice predictions to match y_test length
y_pred_xgb = y_pred_xgb[:len(y_test)]
y_pred_lstm = y_pred_lstm[:len(y_test)]


def mean_absolute_percentage_error(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
```

```
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100

# Calculate MAPE for XGBoost and LSTM
mape_xgb = mean_absolute_percentage_error(y_test, y_pred_xgb)
mape_lstm = mean_absolute_percentage_error(y_test, y_pred_lstm)

print(f"XGBoost MAPE: {mape_xgb}%")
print(f"LSTM MAPE: {mape_lstm}%")
```

```
XGBoost MAPE: 30.37483736866819%
LSTM MAPE: 22.286891169408364%
```

In [ ]:

In [61]:
```
import matplotlib.pyplot as plt

# Residuals for XGBoost and LSTM
residuals_xgb = y_test - y_pred_xgb
residuals_lstm = y_test - y_pred_lstm

# Plot residuals
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.scatter(y_pred_xgb, residuals_xgb)
plt.axhline(y=0, color='r', linestyle='--')
plt.title('XGBoost Residuals')

plt.subplot(1, 2, 2)
plt.scatter(y_pred_lstm, residuals_lstm)
plt.axhline(y=0, color='r', linestyle='--')
plt.title('LSTM Residuals')

plt.show()
```



In [105…
```
from sklearn.model_selection import TimeSeriesSplit
```

```
# Time Series Cross-Validation
tscv = TimeSeriesSplit(n_splits=5)
for train_index, test_index in tscv.split(X):
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    xgb_model.fit(X_train, y_train)
    y_pred = xgb_model.predict(X_test)
    print("XGBoost RMSE:", np.sqrt(mean_squared_error(y_test, y_pred)))
```

```
XGBoost RMSE: 9.258133018697201
XGBoost RMSE: 17.82805792887585
XGBoost RMSE: 8.916977685163804
XGBoost RMSE: 8.742251183329556
XGBoost RMSE: 8.815924690153071
```

In [106…

```
summary = {
    "Model": ["XGBoost", "LSTM"],
    "RMSE": [rmse_xgb.mean(), rmse_lstm.mean()],
    "MAE": [mae_xgb.mean(), mae_lstm.mean()],
    "MAPE": [mape_xgb, mape_lstm],
}

results_df = pd.DataFrame(summary)
print(results_df)
```

```
    Model      RMSE       MAE       MAPE
0  XGBoost  8.614315  6.924497  30.374837
1     LSTM  9.410982  7.572233  22.286891
```

In [ ]:

In [94]:

```
import pandas as pd
import numpy as np
import xgboost as xgb
import matplotlib.pyplot as plt

# Assuming you already have a trained XGBoost model (xgb_model)

# Step 1: Preprocessing function to convert columns to correct data types
def preprocess_data(data):
    # Convert 'Is_Holiday' to boolean (True/False)
    data['Is_Holiday'] = data['Is_Holiday'].astype(bool)

    # Convert price columns to numeric (in case any were objects)
    price_columns = ['Price_Chicken', 'Price_FriedSnacks', 'Price_FriedBurritos', '
    for col in price_columns:
        data[col] = pd.to_numeric(data[col], errors='coerce')

    # Ensure all columns are of the correct type
    data = data.apply(pd.to_numeric, errors='coerce')

    # Fill any missing values that may have been caused by type conversion
    data.fillna(0, inplace=True)

    return data
```

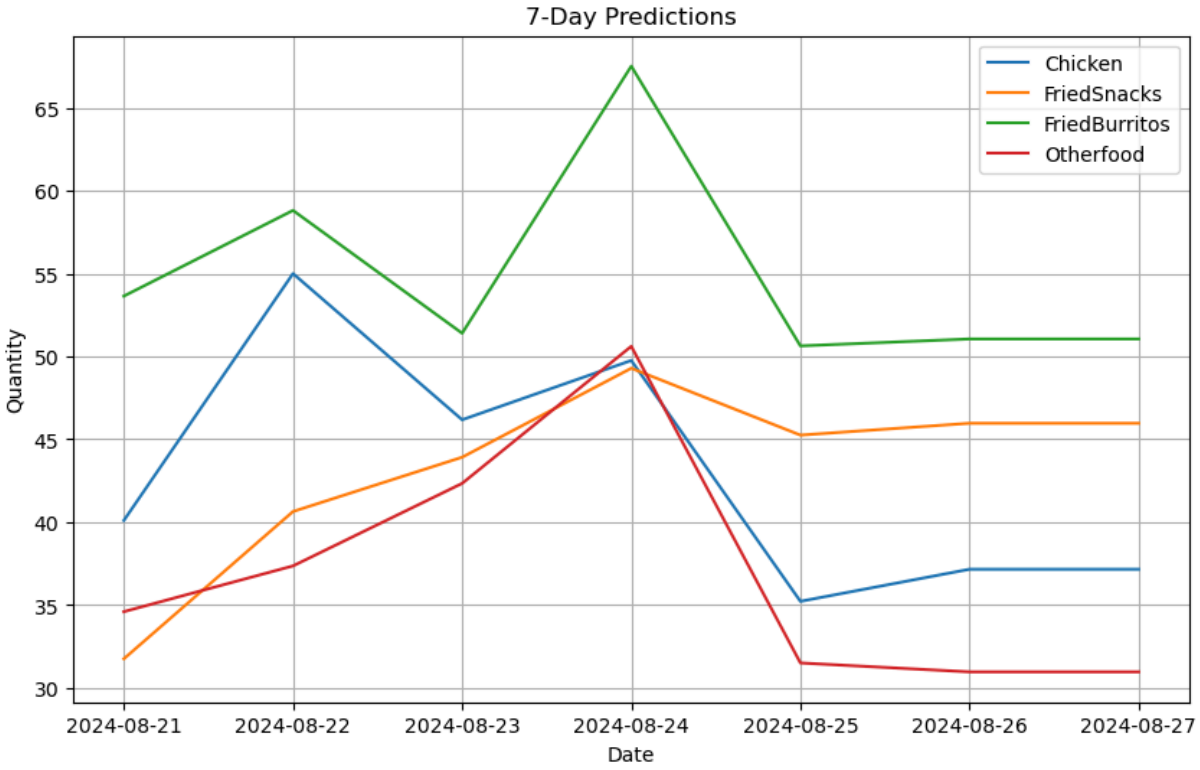
```
# Step 2: Update lag and moving average features after prediction
def update_lag_and_ma(data, predicted_values):
    # Update lag features with the predicted values
    data['Lag_Chicken_Wings'] = predicted_values[0]
    data['Lag_Breakfast'] = predicted_values[1]
    data['Lag_Fried_Burritos'] = predicted_values[2]
    data['Lag_Other_Food'] = predicted_values[3]

    # Update moving averages (simplified, may require a rolling window for your cas
    data['MA_Chicken_Wings'] = np.mean([data['Lag_Chicken_Wings']])  # Example, adj
    data['MA_Breakfast'] = np.mean([data['Lag_Breakfast']])
    data['MA_Fried_Burritos'] = np.mean([data['Lag_Fried_Burritos']])
    data['MA_Other_Food'] = np.mean([data['Lag_Other_Food']])

    return data

# Step 3: Initialize variables
n_days = 7  # Set the number of days you want to predict
future_predictions = []
current_data = df.iloc[-1:].copy()  # Start with the last row of your dataset

# Step 4: Loop to predict for the next n days
for day in range(n_days):
    # Preprocess the data for the current day to ensure correct data types
    current_data = preprocess_data(current_data)

    # Drop target columns to prepare features for prediction
    X_future = current_data.drop(['Chicken', 'FriedSnacks', 'FriedBurritos', 'Other

    # Predict next day's values
    y_future = xgb_model.predict(X_future)

    # Append predictions to the list
    future_predictions.append(y_future[0])

    # Update the features (lags, moving averages) with the predicted values
    current_data = update_lag_and_ma(current_data, y_future[0])

    # Update 'current_data' for the next day's prediction (using the new predicted
    new_row = current_data.copy()

    # Combine features and predictions to simulate the next day
    new_row.iloc[0, :] = np.append(X_future.values[0], y_future[0])
    current_data = new_row

# Step 5: Convert predictions into a DataFrame
future_df = pd.DataFrame(
    future_predictions,
    columns=['Chicken', 'FriedSnacks', 'FriedBurritos', 'Otherfood']
)

# Step 6: Generate future dates (next n days)
future_dates = pd.date_range(start=df.index[-1] + pd.Timedelta(days=1), periods=n_d

# Add the dates to the predictions
future_df['Date'] = future_dates
```

```
future_df.set_index('Date', inplace=True)

# Step 7: Plot the predictions
plt.figure(figsize=(10, 6))
plt.plot(future_df.index, future_df['Chicken'], label='Chicken')
plt.plot(future_df.index, future_df['FriedSnacks'], label='FriedSnacks')
plt.plot(future_df.index, future_df['FriedBurritos'], label='FriedBurritos')
plt.plot(future_df.index, future_df['Otherfood'], label='Otherfood')
plt.legend()
plt.title(f'{n_days}-Day Predictions')
plt.xlabel('Date')
plt.ylabel('Quantity')
plt.grid(True)
plt.show()

# Print the predictions
print(future_df)
```

```
C:\Users\harsh\AppData\Local\Temp\ipykernel_20232\1985423534.py:68: FutureWarning: S
etting an item of incompatible dtype is deprecated and will raise an error in a futu
re version of pandas. Value 'False' has dtype incompatible with int64, please explic
itly cast to a compatible dtype first.
  new_row.iloc[0, :] = np.append(X_future.values[0], y_future[0])
C:\Users\harsh\AppData\Local\Temp\ipykernel_20232\1985423534.py:68: FutureWarning: S
etting an item of incompatible dtype is deprecated and will raise an error in a futu
re version of pandas. Value '1.173000521325178' has dtype incompatible with int64, p
lease explicitly cast to a compatible dtype first.
  new_row.iloc[0, :] = np.append(X_future.values[0], y_future[0])
C:\Users\harsh\AppData\Local\Temp\ipykernel_20232\1985423534.py:68: FutureWarning: S
etting an item of incompatible dtype is deprecated and will raise an error in a futu
re version of pandas. Value '1.1183233945075022' has dtype incompatible with int64,
please explicitly cast to a compatible dtype first.
  new_row.iloc[0, :] = np.append(X_future.values[0], y_future[0])
C:\Users\harsh\AppData\Local\Temp\ipykernel_20232\1985423534.py:68: FutureWarning: S
etting an item of incompatible dtype is deprecated and will raise an error in a futu
re version of pandas. Value '1.1296730673126847' has dtype incompatible with int64,
please explicitly cast to a compatible dtype first.
  new_row.iloc[0, :] = np.append(X_future.values[0], y_future[0])
C:\Users\harsh\AppData\Local\Temp\ipykernel_20232\1985423534.py:68: FutureWarning: S
etting an item of incompatible dtype is deprecated and will raise an error in a futu
re version of pandas. Value '0.9869134765542429' has dtype incompatible with bool, p
lease explicitly cast to a compatible dtype first.
  new_row.iloc[0, :] = np.append(X_future.values[0], y_future[0])
C:\Users\harsh\AppData\Local\Temp\ipykernel_20232\1985423534.py:68: FutureWarning: S
etting an item of incompatible dtype is deprecated and will raise an error in a futu
re version of pandas. Value 'False' has dtype incompatible with float64, please expl
icitly cast to a compatible dtype first.
  new_row.iloc[0, :] = np.append(X_future.values[0], y_future[0])
C:\Users\harsh\AppData\Local\Temp\ipykernel_20232\1985423534.py:68: FutureWarning: S
etting an item of incompatible dtype is deprecated and will raise an error in a futu
re version of pandas. Value '-0.06340976022588453' has dtype incompatible with int3
2, please explicitly cast to a compatible dtype first.
  new_row.iloc[0, :] = np.append(X_future.values[0], y_future[0])
C:\Users\harsh\AppData\Local\Temp\ipykernel_20232\1985423534.py:68: FutureWarning: S
etting an item of incompatible dtype is deprecated and will raise an error in a futu
re version of pandas. Value '-1.241316431398077' has dtype incompatible with int32,
please explicitly cast to a compatible dtype first.
  new_row.iloc[0, :] = np.append(X_future.values[0], y_future[0])
C:\Users\harsh\AppData\Local\Temp\ipykernel_20232\1985423534.py:68: FutureWarning: S
etting an item of incompatible dtype is deprecated and will raise an error in a futu
re version of pandas. Value '0.7177211857401962' has dtype incompatible with bool, p
lease explicitly cast to a compatible dtype first.
  new_row.iloc[0, :] = np.append(X_future.values[0], y_future[0])
C:\Users\harsh\AppData\Local\Temp\ipykernel_20232\1985423534.py:68: FutureWarning: S
etting an item of incompatible dtype is deprecated and will raise an error in a futu
re version of pandas. Value '-0.8132033185439302' has dtype incompatible with int32,
please explicitly cast to a compatible dtype first.
  new_row.iloc[0, :] = np.append(X_future.values[0], y_future[0])
C:\Users\harsh\AppData\Local\Temp\ipykernel_20232\1985423534.py:68: FutureWarning: S
etting an item of incompatible dtype is deprecated and will raise an error in a futu
re version of pandas. Value '0.5674983902312255' has dtype incompatible with float3
2, please explicitly cast to a compatible dtype first.
  new_row.iloc[0, :] = np.append(X_future.values[0], y_future[0])
C:\Users\harsh\AppData\Local\Temp\ipykernel_20232\1985423534.py:68: FutureWarning: S
```

```
etting an item of incompatible dtype is deprecated and will raise an error in a futu
re version of pandas. Value '0.31974356152227207' has dtype incompatible with float3
2, please explicitly cast to a compatible dtype first.
  new_row.iloc[0, :] = np.append(X_future.values[0], y_future[0])
C:\Users\harsh\AppData\Local\Temp\ipykernel_20232\1985423534.py:68: FutureWarning: S
etting an item of incompatible dtype is deprecated and will raise an error in a futu
re version of pandas. Value '-0.07029306155045145' has dtype incompatible with float
32, please explicitly cast to a compatible dtype first.
  new_row.iloc[0, :] = np.append(X_future.values[0], y_future[0])
C:\Users\harsh\AppData\Local\Temp\ipykernel_20232\1985423534.py:68: FutureWarning: S
etting an item of incompatible dtype is deprecated and will raise an error in a futu
re version of pandas. Value '0.40772968168039486' has dtype incompatible with float3
2, please explicitly cast to a compatible dtype first.
  new_row.iloc[0, :] = np.append(X_future.values[0], y_future[0])
C:\Users\harsh\AppData\Local\Temp\ipykernel_20232\1985423534.py:68: FutureWarning: S
etting an item of incompatible dtype is deprecated and will raise an error in a futu
re version of pandas. Value '0.9869134765542429' has dtype incompatible with bool, p
lease explicitly cast to a compatible dtype first.
  new_row.iloc[0, :] = np.append(X_future.values[0], y_future[0])
C:\Users\harsh\AppData\Local\Temp\ipykernel_20232\1985423534.py:68: FutureWarning: S
etting an item of incompatible dtype is deprecated and will raise an error in a futu
re version of pandas. Value 'False' has dtype incompatible with float64, please expl
icitly cast to a compatible dtype first.
  new_row.iloc[0, :] = np.append(X_future.values[0], y_future[0])
C:\Users\harsh\AppData\Local\Temp\ipykernel_20232\1985423534.py:68: FutureWarning: S
etting an item of incompatible dtype is deprecated and will raise an error in a futu
re version of pandas. Value '-0.8132033185439302' has dtype incompatible with int32,
please explicitly cast to a compatible dtype first.
  new_row.iloc[0, :] = np.append(X_future.values[0], y_future[0])
C:\Users\harsh\AppData\Local\Temp\ipykernel_20232\1985423534.py:68: FutureWarning: S
etting an item of incompatible dtype is deprecated and will raise an error in a futu
re version of pandas. Value '-0.07029306155045145' has dtype incompatible with bool,
please explicitly cast to a compatible dtype first.
  new_row.iloc[0, :] = np.append(X_future.values[0], y_future[0])
C:\Users\harsh\AppData\Local\Temp\ipykernel_20232\1985423534.py:68: FutureWarning: S
etting an item of incompatible dtype is deprecated and will raise an error in a futu
re version of pandas. Value 'False' has dtype incompatible with float64, please expl
icitly cast to a compatible dtype first.
  new_row.iloc[0, :] = np.append(X_future.values[0], y_future[0])
C:\Users\harsh\AppData\Local\Temp\ipykernel_20232\1985423534.py:68: FutureWarning: S
etting an item of incompatible dtype is deprecated and will raise an error in a futu
re version of pandas. Value '53.64191818237305' has dtype incompatible with bool, pl
ease explicitly cast to a compatible dtype first.
  new_row.iloc[0, :] = np.append(X_future.values[0], y_future[0])
C:\Users\harsh\AppData\Local\Temp\ipykernel_20232\1985423534.py:68: FutureWarning: S
etting an item of incompatible dtype is deprecated and will raise an error in a futu
re version of pandas. Value '58.8100700378418' has dtype incompatible with bool, ple
ase explicitly cast to a compatible dtype first.
  new_row.iloc[0, :] = np.append(X_future.values[0], y_future[0])
C:\Users\harsh\AppData\Local\Temp\ipykernel_20232\1985423534.py:68: FutureWarning: S
etting an item of incompatible dtype is deprecated and will raise an error in a futu
re version of pandas. Value '51.399410247802734' has dtype incompatible with bool, p
lease explicitly cast to a compatible dtype first.
  new_row.iloc[0, :] = np.append(X_future.values[0], y_future[0])
C:\Users\harsh\AppData\Local\Temp\ipykernel_20232\1985423534.py:68: FutureWarning: S
etting an item of incompatible dtype is deprecated and will raise an error in a futu
```

```
re version of pandas. Value '67.51419067382812' has dtype incompatible with bool, pl
ease explicitly cast to a compatible dtype first.
  new_row.iloc[0, :] = np.append(X_future.values[0], y_future[0])
C:\Users\harsh\AppData\Local\Temp\ipykernel_20232\1985423534.py:68: FutureWarning: S
etting an item of incompatible dtype is deprecated and will raise an error in a futu
re version of pandas. Value '50.63688278198242' has dtype incompatible with bool, pl
ease explicitly cast to a compatible dtype first.
  new_row.iloc[0, :] = np.append(X_future.values[0], y_future[0])
```



Date	Chicken	FriedSnacks	FriedBurritos	Otherfood
2024-08-21	40.107643	31.758524	53.641918	34.603230
2024-08-22	55.000347	40.643997	58.810070	37.362503
2024-08-23	46.179920	43.919964	51.399410	42.341339
2024-08-24	49.757332	49.288071	67.514191	50.611992
2024-08-25	35.226627	45.254295	50.636883	31.508507
2024-08-26	37.161419	45.969299	51.055069	30.970245
2024-08-27	37.161419	45.969299	51.055069	30.970245

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: