Final Report

November 25, 2022

1 2D Design Template

2 Overview

The purpose of this project is for you to apply what you have learnt in this course. This includes working with data and visualizing it, create model of linear regression or logistic regression, as well as using metrics to measure the accuracy of your model.

Please find the project handout description in the following link: - DDW-MU-Humanities Handout - DDW-MU-SocialStudies Handout

2.1 Deliverables

You need to submit this Jupyter notebook together with the dataset into Vocareum. Use the template in this notebook to work on this project.

2.2 Students Submission

Student's Name: - Bundhoo Simriti - Elvern Neylmav Tanny - Koh Chee Kiat - Haritha Shraeya Rajasekar - Mahima Sharma - Zhang Jianyu

```
In [1]: # Import Libraries
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
        import numpy as np
```

2.3 Overview About the Problem

Describe here the problem you are trying to solve.

2.3.1 Problem Statement

We aim to predict the future of food safety and security in countries of the lower income bracket like Cambodia and Myanmar through the prediction of amount of undernourishment as it encapsulates percentage of people whose food requirements are not satisfied.

- Southeast Asia is a diverse, fast-growing region, making remarkable progress in terms of improving food security, going from 31% undernourishment in the 1990s to below 10% by 2014-2016 (https://www.fao.org/3/bt099e/bt099e.pdf). Although undernourishment has been decreasing, food security is still a concern when accounting for the fast-growing population in Southeast Asia, projected to grow from 640 million to more than 710 million by 2030 (https://www.enterprisesg.gov.sg/overseas-markets/asia-pacific/asean/overview). Therefore, our group focused on modelling the food security of Southeast Asia.

2.4 Dataset

Describe here your data set. Put the link to the sources of your dataset. Describe your data and what are the columns.

Put some Python codes here to describe and visualize your data.

Description of Dataset: The GDP per capita dataset documents gross domestic product per person in USD of a country since 1960.

Inflation, consumer prices (annual %) documents the increase or decrease in inflation rate in comparison to the previous year since 1960.

Gross per capita Production Index Number (2014-2016 = 100) (Food production index) documents relative index of food production in the country. With the 3 year period 2014-2016 given 100 points as a reference since 2001.

Percentage of Undernourishment Prevalence documents percentage of people whose food requirements aren't satisfied since 2000.

Our model uses 'GDP per Capita (USD)', 'Annual Inflation Rate (%)', 'Food production index (2014-2016 = 100)' as features and 'Percentage of Undernourishment Prevalence' as target spanning from 2001 to 2020

Sources: GDP per capita dataset (USD) - https://data.worldbank.org/indicator/NY.GDP.PCAP.CD

Inflation, consumer prices (annual %) - https://data.worldbank.org/indicator/FP.CPI.TOTL.ZG

Gross per capita Production Index Number (2014-2016 = 100) (Food production index) - https://www.fao.org/faostat/en/#data/QI

Percentage of Undernourishment Prevalence - https://www.fao.org/faostat/en/#data/FS

Dataset iterations: We first focused on all countries in Southeast Asia and then worked our way to decreasing the number of countries as the data range was too diverse and hence meaningful conclusions could not be achieved. We focussed our scope to emphasize on countries with lower-income and having agricultural significance, specifically Myanmar and Cambodia.

We decided to narrow our scope to lower-middle income countries categorised by the World Bank (https://datatopics.worldbank.org/world-development-indicators/the-world-by-income-and-region.html) and focus on countries whose economies are more dependent on agriculture as such countries are more at risk of facing food security issues. This left us with Cambodia and Myanmar, where agriculture makes up more than 20% of their GDP (22.7% and 20.9% respectively, 2020) https://data.worldbank.org/indicator/NV.AGR.TOTL.ZS?end=2020&locations=KH-MM&start=2020&view=bar

We explored multiple categories of features and performed linear regression on them one at a time and identified those features with which we were able to identify a relation: economic, environmental, food production features. After testing them, we came to a conclusion to use the following economic measures 'GDP per Capita (USD)', 'Annual Inflation Rate (%)', 'Food production index (2014-2016 = 100)' as our features. We chose prevalence of undernourishment as a (%) as our target.

2.5 Cleaning Dataset

2.5.1 Percentage of Undernourishment Prevalence

In [2]:
```
# Read the Percentage of Undernourishment Prevalence CSV file online
df_pup_url = "https://raw.githubusercontent.com/verneylmavt/2D_Project_Term-3/main/2D_I
df_pup = pd.read_csv(df_pup_url, encoding='latin-1')
df_pup = df_pup.set_index("Area")
display(df_pup)
```

```
            ï»¿Domain Code                           Domain  \
Area
Afghanistan            FS  Suite of Food Security Indicators
Afghanistan            FS  Suite of Food Security Indicators
Afghanistan            FS  Suite of Food Security Indicators
Afghanistan            FS  Suite of Food Security Indicators
Afghanistan            FS  Suite of Food Security Indicators
...                   ...                                ...
Zimbabwe               FS  Suite of Food Security Indicators
Zimbabwe               FS  Suite of Food Security Indicators
Zimbabwe               FS  Suite of Food Security Indicators
Zimbabwe               FS  Suite of Food Security Indicators
Zimbabwe               FS  Suite of Food Security Indicators

             Area Code (M49)  Element Code Element  Item Code  \
Area
Afghanistan                4          6121   Value     210041
Afghanistan                4          6121   Value     210041
Afghanistan                4          6121   Value     210041
Afghanistan                4          6121   Value     210041
Afghanistan                4          6121   Value     210041
...                      ...           ...     ...        ...
Zimbabwe                 716          6121   Value     210041
Zimbabwe                 716          6121   Value     210041
Zimbabwe                 716          6121   Value     210041
Zimbabwe                 716          6121   Value     210041
Zimbabwe                 716          6121   Value     210041

                                                          Item  Year Code  \
Area
Afghanistan  Prevalence of undernourishment (percent) (3-ye...   20002002
Afghanistan  Prevalence of undernourishment (percent) (3-ye...   20012003
Afghanistan  Prevalence of undernourishment (percent) (3-ye...   20022004
Afghanistan  Prevalence of undernourishment (percent) (3-ye...   20032005
```

```
Afghanistan  Prevalence of undernourishment (percent) (3-ye...   20042006
...                                                     ...        ...
Zimbabwe     Prevalence of undernourishment (percent) (3-ye...   20152017
Zimbabwe     Prevalence of undernourishment (percent) (3-ye...   20162018
Zimbabwe     Prevalence of undernourishment (percent) (3-ye...   20172019
Zimbabwe     Prevalence of undernourishment (percent) (3-ye...   20182020
Zimbabwe     Prevalence of undernourishment (percent) (3-ye...   20192021

                  Year Unit Value Flag Flag Description  Note
Area
Afghanistan  2000-2002    %  47.8    E  Estimated value   NaN
Afghanistan  2001-2003    %  45.6    E  Estimated value   NaN
Afghanistan  2002-2004    %  40.6    E  Estimated value   NaN
Afghanistan  2003-2005    %    38    E  Estimated value   NaN
Afghanistan  2004-2006    %  36.1    E  Estimated value   NaN
...                ...  ...   ...  ...              ...   ...
Zimbabwe     2015-2017    %   NaN    O    Missing value   NaN
Zimbabwe     2016-2018    %   NaN    O    Missing value   NaN
Zimbabwe     2017-2019    %   NaN    O    Missing value   NaN
Zimbabwe     2018-2020    %   NaN    O    Missing value   NaN
Zimbabwe     2019-2021    %   NaN    O    Missing value   NaN

[4102 rows x 14 columns]
```

```
In [3]: # Extracting data for Myanmar and Cambodia
        df_pup_cambodia_myanmar = (df_pup.loc[["Cambodia", "Myanmar"], :]).copy()
        cambodia_pup = (df_pup_cambodia_myanmar.loc["Cambodia", "Value"]).tolist()
        myanmar_pup = (df_pup_cambodia_myanmar.loc["Myanmar", "Value"]).tolist()
```

2.5.2 GDP per Capita (USD)

```
In [4]: # Read the GDP per capita (USD) CSV file online
        df_gdp_url = "https://raw.githubusercontent.com/verneylmavt/2D_Project_Term-3/main/2D_I
        df_gdp = pd.read_csv(df_gdp_url)
        df_gdp = df_gdp.set_index("Country Name")
        display(df_gdp)
```

```
                            Country Code               Indicator Name  \
Country Name
Aruba                                ABW  GDP per capita (current US$)
Africa Eastern and Southern          AFE  GDP per capita (current US$)
Afghanistan                          AFG  GDP per capita (current US$)
Africa Western and Central           AFW  GDP per capita (current US$)
Angola                               AGO  GDP per capita (current US$)
...                                  ...                           ...
Kosovo                               XKX  GDP per capita (current US$)
Yemen, Rep.                          YEM  GDP per capita (current US$)
```

```
South Africa                       ZAF  GDP per capita (current US$)
Zambia                             ZMB  GDP per capita (current US$)
Zimbabwe                           ZWE  GDP per capita (current US$)

                             Indicator Code        1960        1961  \
Country Name
Aruba                        NY.GDP.PCAP.CD         NaN         NaN
Africa Eastern and Southern  NY.GDP.PCAP.CD  162.726326  162.555968
Afghanistan                  NY.GDP.PCAP.CD   59.773234   59.860900
Africa Western and Central   NY.GDP.PCAP.CD  107.930722  113.080062
Angola                       NY.GDP.PCAP.CD         NaN         NaN
...                                     ...         ...         ...
Kosovo                       NY.GDP.PCAP.CD         NaN         NaN
Yemen, Rep.                  NY.GDP.PCAP.CD         NaN         NaN
South Africa                 NY.GDP.PCAP.CD  511.618737  526.461750
Zambia                       NY.GDP.PCAP.CD  232.188565  220.042067
Zimbabwe                     NY.GDP.PCAP.CD  278.813847  280.828663

                                   1962        1963        1964        1965  \
Country Name
Aruba                               NaN         NaN         NaN         NaN
Africa Eastern and Southern  172.271022  199.784916  180.228774  199.517227
Afghanistan                   58.458009   78.706429   82.095307  101.108325
Africa Western and Central   118.829461  123.441090  131.852423  138.524029
Angola                              NaN         NaN         NaN         NaN
...                                 ...         ...         ...         ...
Kosovo                              NaN         NaN         NaN         NaN
Yemen, Rep.                         NaN         NaN         NaN         NaN
South Africa                 546.261935  589.160460  632.716104  674.186433
Zambia                       212.578449  213.896759  242.384472  303.281741
Zimbabwe                     276.688233  277.479715  281.558896  293.308788

                                   1966  ...          2012          2013  \
Country Name                             ...
Aruba                               NaN  ...  25496.843940  26442.426800
Africa Eastern and Southern  211.054388  ...   1777.303950   1748.905594
Afghanistan                  137.594298  ...    638.845852    624.315454
Africa Western and Central   144.323882  ...   1965.115750   2157.494584
Angola                              NaN  ...   4978.434435   5127.717243
...                                 ...  ...           ...           ...
Kosovo                              NaN  ...   3410.859780   3704.784221
Yemen, Rep.                         NaN  ...   1446.536472   1607.152173
South Africa                 714.562010  ...   8222.197279   7467.079185
Zambia                       343.373670  ...   1763.069442   1878.346811
Zimbabwe                     277.234532  ...   1304.968011   1429.998461

                                     2014          2015          2016  \
Country Name
```

```
Aruba                         26895.057170  28399.050130  28453.715560
Africa Eastern and Southern    1736.242220   1556.316469   1446.533624
Afghanistan                     614.223342    556.007221    512.012778
Africa Western and Central     2212.914095   1894.322115   1673.843681
Angola                         5094.112329   3127.890598   1728.023754
...                                    ...           ...           ...
Kosovo                         3902.676013   3520.766449   3759.560246
Yemen, Rep.                    1674.002572   1601.807163   1152.738019
South Africa                   6988.808739   6259.839681   5756.965741
Zambia                         1762.427817   1338.290927   1280.806543
Zimbabwe                       1434.896277   1445.069702   1464.588957

                                      2017          2018          2019  \
Country Name
Aruba                         29348.418970  30253.714230  31135.884360
Africa Eastern and Southern    1629.404273   1541.031661   1511.309259
Afghanistan                     516.679862    485.668419    494.179350
Africa Western and Central     1613.490478   1704.135698   1777.852822
Angola                         2313.220584   2524.942483   2177.799015
...                                    ...           ...           ...
Kosovo                         4009.380987   4384.048892   4416.108358
Yemen, Rep.                     964.340344    758.145949    750.554583
South Africa                   6690.939847   7005.095413   6624.761865
Zambia                         1535.196574   1516.368371   1305.001031
Zimbabwe                       1235.189032   1254.642265   1316.740657

                                      2020         2021
Country Name
Aruba                         23384.298790          NaN
Africa Eastern and Southern    1360.878645  1557.722682
Afghanistan                     516.747871          NaN
Africa Western and Central     1709.764129  1774.921218
Angola                         1631.431691  2137.909393
...                                    ...          ...
Kosovo                         4310.811183  4986.582469
Yemen, Rep.                     631.681490   690.759273
South Africa                   5655.867654  6994.211654
Zambia                          985.132436  1120.630171
Zimbabwe                       1214.509820  1737.173977

[266 rows x 65 columns]
```

In [5]:
```python
# Extracting data for Myanmar and Cambodia for required year range
df_gdp_cambodia_myanmar = (df_gdp.loc[["Cambodia", "Myanmar"], "2001":"2020"]).copy()
cambodia_gdp = (df_gdp_cambodia_myanmar.loc["Cambodia", "2001":"2020"]).tolist()
myanmar_gdp = (df_gdp_cambodia_myanmar.loc["Myanmar", "2001":"2020"]).tolist()
```

2.5.3 Annual Inflation Rate (%)

```
In [6]: # Read the Annual Inflation Rate (%) CSV file online
        df_inflation_rate_url = "https://raw.githubusercontent.com/verneylmavt/2D_Project_Term-
        df_inflation_rate = pd.read_csv(df_inflation_rate_url)
        df_inflation_rate = df_inflation_rate.set_index("Country Name")
        display(df_inflation_rate)
```

```
                             Country Code  \
Country Name
Aruba                                 ABW
Africa Eastern and Southern           AFE
Afghanistan                           AFG
Africa Western and Central            AFW
Angola                                AGO
...                                   ...
Kosovo                                XKX
Yemen, Rep.                           YEM
South Africa                          ZAF
Zambia                                ZMB
Zimbabwe                              ZWE

                                                    Indicator Name  \
Country Name
Aruba                        Inflation, consumer prices (annual %)
Africa Eastern and Southern  Inflation, consumer prices (annual %)
Afghanistan                  Inflation, consumer prices (annual %)
Africa Western and Central   Inflation, consumer prices (annual %)
Angola                       Inflation, consumer prices (annual %)
...                                                            ...
Kosovo                       Inflation, consumer prices (annual %)
Yemen, Rep.                  Inflation, consumer prices (annual %)
South Africa                 Inflation, consumer prices (annual %)
Zambia                       Inflation, consumer prices (annual %)
Zimbabwe                     Inflation, consumer prices (annual %)

                             Indicator Code      1960      1961      1962  \
Country Name
Aruba                        FP.CPI.TOTL.ZG       NaN       NaN       NaN
Africa Eastern and Southern  FP.CPI.TOTL.ZG       NaN       NaN       NaN
Afghanistan                  FP.CPI.TOTL.ZG       NaN       NaN       NaN
Africa Western and Central   FP.CPI.TOTL.ZG       NaN       NaN       NaN
Angola                       FP.CPI.TOTL.ZG       NaN       NaN       NaN
...                                     ...       ...       ...       ...
Kosovo                       FP.CPI.TOTL.ZG       NaN       NaN       NaN
Yemen, Rep.                  FP.CPI.TOTL.ZG       NaN       NaN       NaN
South Africa                 FP.CPI.TOTL.ZG  1.288859  2.102374  1.246285
Zambia                       FP.CPI.TOTL.ZG       NaN       NaN       NaN
Zimbabwe                     FP.CPI.TOTL.ZG       NaN       NaN       NaN
```

```
                               1963         1964      1965      1966  ...  \
Country Name                                                          ...
Aruba                           NaN          NaN       NaN       NaN  ...
Africa Eastern and Southern     NaN          NaN       NaN       NaN  ...
Afghanistan                     NaN          NaN       NaN       NaN  ...
Africa Western and Central      NaN          NaN       NaN       NaN  ...
Angola                          NaN          NaN       NaN       NaN  ...
...                             ...          ...       ...       ...  ...
Kosovo                          NaN          NaN       NaN       NaN  ...
Yemen, Rep.                     NaN          NaN       NaN       NaN  ...
South Africa                1.33797  2.534972841  4.069029  3.489234  ...
Zambia                          NaN          NaN       NaN       NaN  ...
Zimbabwe                        NaN          NaN       NaN       NaN  ...

                                  2012       2013      2014       2015  \
Country Name
Aruba                         0.627472  -2.372065  0.421441   0.474764
Africa Eastern and Southern   9.158707   5.750981  5.370290   5.250171
Afghanistan                   6.441213   7.385772  4.673996  -0.661709
Africa Western and Central    4.578375   2.439201  1.758052   2.130268
Angola                       10.277905   8.777814  7.280387   9.150372
...                                ...        ...       ...        ...
Kosovo                        2.476738   1.767324  0.428958  -0.536929
Yemen, Rep.                   9.885387  10.968442  8.104726        NaN
South Africa                  5.724658   5.784469  6.129838   4.540642
Zambia                        6.575900   6.977676  7.806876  10.110593
Zimbabwe                      3.725327   1.634950 -0.197785  -2.430968

                                  2016       2017       2018        2019  \
Country Name
Aruba                        -0.931196  -1.028282   3.626041    4.257462
Africa Eastern and Southern   6.571396   6.399343   4.720811    4.120246
Afghanistan                   4.383892   4.975952   0.626149    2.302373
Africa Western and Central    1.494564   1.764635   1.784050    1.758565
Angola                       30.695313  29.843587  19.628608   17.081215
...                                ...        ...        ...         ...
Kosovo                        0.273169   1.488234   1.053798    2.675992
Yemen, Rep.                        NaN        NaN        NaN         NaN
South Africa                  6.571396   5.184247   4.517165    4.120246
Zambia                       17.869730   6.577312   7.494572    9.150316
Zimbabwe                     -1.543670   0.893962  10.618866  255.304991

                                  2020       2021
Country Name
Aruba                              NaN        NaN
Africa Eastern and Southern   5.404815   7.240978
Afghanistan                        NaN        NaN
```

```
Africa Western and Central     2.492522   3.925603
Angola                              NaN        NaN
...                                 ...        ...
Kosovo                         0.198228   3.353691
Yemen, Rep.                         NaN        NaN
South Africa                   3.210036   4.611672
Zambia                        15.732585  22.021234
Zimbabwe                     557.201817  98.546105

[266 rows x 65 columns]
```

```
In [7]: # Extracting data for Myanmar and Cambodia for required year range
        df_inflation_rate_cambodia_myanmar = (df_inflation_rate.loc[["Cambodia", "Myanmar"], "
        cambodia_inflation_rate = (df_inflation_rate_cambodia_myanmar.loc["Cambodia", "2001":"
        myanmar_inflation_rate = (df_inflation_rate_cambodia_myanmar.loc["Myanmar", "2001":"20
```

2.5.4 Gross per capita Production Index Number (2014-2016 = 100)

```
In [8]: # Read the Production Index Number CSV file online
        df_pin1416_url = "https://raw.githubusercontent.com/verneylmavt/2D_Project_Term-3/main
        df_pin1416 = pd.read_csv(df_pin1416_url, encoding='latin-1')
        df_pin1416 = df_pin1416.set_index("Area")
        display(df_pin1416)
```

```
            ï»¿Domain Code             Domain  Area Code (M49)  Element Code  \
Area
Afghanistan            QI  Production Indices                4           434
Afghanistan            QI  Production Indices                4           434
Afghanistan            QI  Production Indices                4           434
Afghanistan            QI  Production Indices                4           434
Afghanistan            QI  Production Indices                4           434
...                   ...                 ...              ...           ...
Zimbabwe               QI  Production Indices              716           434
Zimbabwe               QI  Production Indices              716           434
Zimbabwe               QI  Production Indices              716           434
Zimbabwe               QI  Production Indices              716           434
Zimbabwe               QI  Production Indices              716           434

                                                        Element  \
Area
Afghanistan  Gross per capita Production Index Number (2014...
Afghanistan  Gross per capita Production Index Number (2014...
Afghanistan  Gross per capita Production Index Number (2014...
Afghanistan  Gross per capita Production Index Number (2014...
Afghanistan  Gross per capita Production Index Number (2014...
...                                                        ...
Zimbabwe     Gross per capita Production Index Number (2014...
```

```
Zimbabwe     Gross per capita Production Index Number (2014...
Zimbabwe     Gross per capita Production Index Number (2014...
Zimbabwe     Gross per capita Production Index Number (2014...
Zimbabwe     Gross per capita Production Index Number (2014...

            Item Code (CPC)        Item  Year Code  Year   Unit   Value Flag  \
Area
Afghanistan           F2051  Agriculture       1961  1961  index  161.93    E
Afghanistan           F2051  Agriculture       1962  1962  index  161.58    E
Afghanistan           F2051  Agriculture       1963  1963  index  159.56    E
Afghanistan           F2051  Agriculture       1964  1964  index  166.81    E
Afghanistan           F2051  Agriculture       1965  1965  index  170.37    E
...                     ...          ...        ...   ...    ...     ...  ...
Zimbabwe              F2051  Agriculture       2016  2016  index   95.59    E
Zimbabwe              F2051  Agriculture       2017  2017  index  100.70    E
Zimbabwe              F2051  Agriculture       2018  2018  index  115.99    E
Zimbabwe              F2051  Agriculture       2019  2019  index   91.59    E
Zimbabwe              F2051  Agriculture       2020  2020  index  105.22    E

            Flag Description
Area
Afghanistan  Estimated value
Afghanistan  Estimated value
Afghanistan  Estimated value
Afghanistan  Estimated value
Afghanistan  Estimated value
...                      ...
Zimbabwe     Estimated value
Zimbabwe     Estimated value
Zimbabwe     Estimated value
Zimbabwe     Estimated value
Zimbabwe     Estimated value

[10920 rows x 13 columns]
```

```
In [9]: # Extracting data for Myanmar and Cambodia for required year range
        df_pin1416_cambodia_myanmar = (df_pin1416.loc[["Cambodia", "Myanmar"], :]).copy()
        df_pin1416_cambodia_myanmar["Year Code"] = df_pin1416_cambodia_myanmar["Year Code"].ast
        df_pin1416_cambodia_myanmar = df_pin1416_cambodia_myanmar.loc[
            (df_pin1416_cambodia_myanmar["Year Code"] >= 2001) & (df_pin1416_cambodia_myanmar["

        cambodia_pin1416 = (df_pin1416_cambodia_myanmar.loc["Cambodia", "Value"]).tolist()
        myanmar_pin1416 = (df_pin1416_cambodia_myanmar.loc["Myanmar", "Value"]).tolist()
```

2.6 Combining Relevant Data Extracted into DataFrame

Note: With datasets using three year average, we considered the central year. Eg: 2000-2002 => 2001

```
In [10]: #Instantiating a new DataFrame called df, and adding a new column correspondingly
         df = pd.DataFrame()
         df["Country"] = pd.concat([pd.DataFrame(np.full((20,), "Cambodia")), pd.DataFrame(np.f
         df["Year"] = df_pin1416_cambodia_myanmar.loc[:, "Year Code"].tolist()
         df["Percentage of Undernourishment Prevalence (3-Year Average)"] = cambodia_pup + myan
         df["Binary Categorical"] = pd.concat([pd.DataFrame(np.full((20,), 0)), pd.DataFrame(np
         df["GDP per Capita (USD)"] = cambodia_gdp + myanmar_gdp
         df["Annual Inflation Rate (%)"] = cambodia_inflation_rate + myanmar_inflation_rate
         df["Food Production Index (2014-2016 = 100)"] = cambodia_pin1416 + myanmar_pin1416
```

```
In [11]: # 7 Columns: 4 Features & 1 Tragets with 2 additional columns as a Description (Count
         # 40 Rows, 2001-2020 Cambodia (20) & 2001-2020 Myanmar (20)
         display(df)
         print(df.shape)
```

```
     Country  Year Percentage of Undernourishment Prevalence (3-Year Average)  \
0   Cambodia  2001                                               23.6
1   Cambodia  2002                                               21.2
2   Cambodia  2003                                               19.4
3   Cambodia  2004                                               18.5
4   Cambodia  2005                                                 17
5   Cambodia  2006                                               15.6
6   Cambodia  2007                                               14.8
7   Cambodia  2008                                               14.5
8   Cambodia  2009                                                 13
9   Cambodia  2010                                               11.2
10  Cambodia  2011                                                9.7
11  Cambodia  2012                                                9.5
12  Cambodia  2013                                                9.4
13  Cambodia  2014                                                9.2
14  Cambodia  2015                                                8.9
15  Cambodia  2016                                                8.5
16  Cambodia  2017                                                7.7
17  Cambodia  2018                                                6.6
18  Cambodia  2019                                                  6
19  Cambodia  2020                                                6.3
20   Myanmar  2001                                               37.6
21   Myanmar  2002                                               34.8
22   Myanmar  2003                                               32.4
23   Myanmar  2004                                               30.2
24   Myanmar  2005                                               27.8
25   Myanmar  2006                                               24.5
26   Myanmar  2007                                               20.5
27   Myanmar  2008                                               17.1
```

28	Myanmar	2009	12.9
29	Myanmar	2010	10.2
30	Myanmar	2011	7.8
31	Myanmar	2012	7.1
32	Myanmar	2013	6.1
33	Myanmar	2014	5.1
34	Myanmar	2015	4.2
35	Myanmar	2016	3.5
36	Myanmar	2017	3
37	Myanmar	2018	2.6
38	Myanmar	2019	<2.5
39	Myanmar	2020	3.1

	Binary Categorical	GDP per Capita (USD)	Annual Inflation Rate (%) \
0	0	321.150224	-0.600648
1	0	338.987477	0.211467
2	0	362.335482	0.941746
3	0	408.513639	4.319337
4	0	474.111192	6.615259
5	0	539.750329	5.810686
6	0	631.525258	8.708828
7	0	745.609127	24.096852
8	0	738.054731	-1.241718
9	0	785.502667	3.996395
10	0	882.275614	5.478447
11	0	950.880346	2.934316
12	0	1013.420536	2.941625
13	0	1093.495976	3.855689
14	0	1162.904995	1.223932
15	0	1269.591499	3.019140
16	0	1385.260066	2.912636
17	0	1512.126989	2.459085
18	0	1643.121389	1.942575
19	0	1547.511388	2.940295
20	1	131.715298	21.101305
21	1	128.099702	57.074511
22	1	161.055524	36.589718
23	1	193.368766	4.534214
24	1	216.311501	9.368618
25	1	240.624014	19.996487
26	1	314.202294	35.024597
27	1	460.908889	26.799537
28	1	586.168180	1.472343
29	1	746.945360	7.718382
30	1	1061.344429	5.021460
31	1	1134.302224	1.467583
32	1	1168.165453	5.643039
33	1	1210.097654	4.953299

34	1	1196.743333	9.454172
35	1	1136.610627	6.928825
36	1	1151.114464	4.572537
37	1	1250.173685	6.872329
38	1	1271.111536	8.825067
39	1	1450.662673	7.092387

	Food Production Index (2014-2016 = 100)
0	52.79
1	49.46
2	57.65
3	53.83
4	68.71
5	73.50
6	75.90
7	81.68
8	84.33
9	89.31
10	100.67
11	102.56
12	102.11
13	99.42
14	98.42
15	102.17
16	105.05
17	106.81
18	106.21
19	103.61
20	59.21
21	60.64
22	65.22
23	69.95
24	77.51
25	86.08
26	90.42
27	96.56
28	100.13
29	102.32
30	97.60
31	95.65
32	99.23
33	98.77
34	101.00
35	100.22
36	99.96
37	101.29
38	100.82
39	99.54

```
(40, 7)
```

```
In [12]: # Removing < in "Percentage of Undernourishment Prevalence (3-Year Average)" column
         # Because it changes in-place, we add try statements to prevent error
         try:
             df["Percentage of Undernourishment Prevalence (3-Year Average)"] = pd.to_numeric(
         except:
             pass
```

```
In [13]: # Making sure every dataset used are numbers
         df = df.astype({'Percentage of Undernourishment Prevalence (3-Year Average)':'float',
                         'GDP per Capita (USD)':'float',
                         'Annual Inflation Rate (%)': 'float',
                         'Food Production Index (2014-2016 = 100)': 'float'
                         })
```

```
In [14]: display(df)
         print(df.shape)
```

```
     Country  Year  \
0   Cambodia  2001
1   Cambodia  2002
2   Cambodia  2003
3   Cambodia  2004
4   Cambodia  2005
5   Cambodia  2006
6   Cambodia  2007
7   Cambodia  2008
8   Cambodia  2009
9   Cambodia  2010
10  Cambodia  2011
11  Cambodia  2012
12  Cambodia  2013
13  Cambodia  2014
14  Cambodia  2015
15  Cambodia  2016
16  Cambodia  2017
17  Cambodia  2018
18  Cambodia  2019
19  Cambodia  2020
20   Myanmar  2001
21   Myanmar  2002
22   Myanmar  2003
23   Myanmar  2004
24   Myanmar  2005
25   Myanmar  2006
```

```
26   Myanmar  2007
27   Myanmar  2008
28   Myanmar  2009
29   Myanmar  2010
30   Myanmar  2011
31   Myanmar  2012
32   Myanmar  2013
33   Myanmar  2014
34   Myanmar  2015
35   Myanmar  2016
36   Myanmar  2017
37   Myanmar  2018
38   Myanmar  2019
39   Myanmar  2020

    Percentage of Undernourishment Prevalence (3-Year Average)  \
0                                                 23.6
1                                                 21.2
2                                                 19.4
3                                                 18.5
4                                                 17.0
5                                                 15.6
6                                                 14.8
7                                                 14.5
8                                                 13.0
9                                                 11.2
10                                                 9.7
11                                                 9.5
12                                                 9.4
13                                                 9.2
14                                                 8.9
15                                                 8.5
16                                                 7.7
17                                                 6.6
18                                                 6.0
19                                                 6.3
20                                                37.6
21                                                34.8
22                                                32.4
23                                                30.2
24                                                27.8
25                                                24.5
26                                                20.5
27                                                17.1
28                                                12.9
29                                                10.2
30                                                 7.8
31                                                 7.1
```

32	6.1
33	5.1
34	4.2
35	3.5
36	3.0
37	2.6
38	2.5
39	3.1

	Binary Categorical	GDP per Capita (USD)	Annual Inflation Rate (%)	\
0	0	321.150224	-0.600648	
1	0	338.987477	0.211467	
2	0	362.335482	0.941746	
3	0	408.513639	4.319337	
4	0	474.111192	6.615259	
5	0	539.750329	5.810686	
6	0	631.525258	8.708828	
7	0	745.609127	24.096852	
8	0	738.054731	-1.241718	
9	0	785.502667	3.996395	
10	0	882.275614	5.478447	
11	0	950.880346	2.934316	
12	0	1013.420536	2.941625	
13	0	1093.495976	3.855689	
14	0	1162.904995	1.223932	
15	0	1269.591499	3.019140	
16	0	1385.260066	2.912636	
17	0	1512.126989	2.459085	
18	0	1643.121389	1.942575	
19	0	1547.511388	2.940295	
20	1	131.715298	21.101305	
21	1	128.099702	57.074511	
22	1	161.055524	36.589718	
23	1	193.368766	4.534214	
24	1	216.311501	9.368618	
25	1	240.624014	19.996487	
26	1	314.202294	35.024597	
27	1	460.908889	26.799537	
28	1	586.168180	1.472343	
29	1	746.945360	7.718382	
30	1	1061.344429	5.021460	
31	1	1134.302224	1.467583	
32	1	1168.165453	5.643039	
33	1	1210.097654	4.953299	
34	1	1196.743333	9.454172	
35	1	1136.610627	6.928825	
36	1	1151.114464	4.572537	
37	1	1250.173685	6.872329	

38	1	1271.111536	8.825067
39	1	1450.662673	7.092387

	Food Production Index (2014-2016 = 100)
0	52.79
1	49.46
2	57.65
3	53.83
4	68.71
5	73.50
6	75.90
7	81.68
8	84.33
9	89.31
10	100.67
11	102.56
12	102.11
13	99.42
14	98.42
15	102.17
16	105.05
17	106.81
18	106.21
19	103.61
20	59.21
21	60.64
22	65.22
23	69.95
24	77.51
25	86.08
26	90.42
27	96.56
28	100.13
29	102.32
30	97.60
31	95.65
32	99.23
33	98.77
34	101.00
35	100.22
36	99.96
37	101.29
38	100.82
39	99.54

(40, 7)

```
In [15]: # List of all columns
         print(list(df.columns))
```

```
['Country', 'Year', 'Percentage of Undernourishment Prevalence (3-Year Average)', 'Binary Cate
```

2.7 Functions

2.7.1 Preparation Functions

```
In [16]: # Preparation Functions
         def get_features_targets(df, feature_names, target_names):
             df_feature = df.loc[:, feature_names]
             df_target = df.loc[:, target_names]
             return pd.DataFrame(df_feature), pd.DataFrame(df_target)


         def split_data(df_feature, df_target, random_state=None, test_size=0.5):
             df_feature_rows, df_feature_columns = df_feature.shape
             array_all = list(range(0, df_feature_rows))

             np.random.seed(random_state)
             array_test = list(np.random.choice(array_all, int((df_feature_rows)*test_size), re
             array_train = [i for i in array_all if i not in array_test]

             df_feature_test = df_feature.iloc[array_test, :]
             df_feature_train = df_feature.iloc[array_train, :]
             df_target_test = df_target.iloc[array_test, :]
             df_target_train = df_target.iloc[array_train, :]

             return df_feature_train, df_feature_test, df_target_train, df_target_test


         def normalize_z(dfin):
             mean = dfin.mean(axis=0)
             sd = dfin.std(axis=0)
             dfout = ((dfin.copy())-mean)/sd
             return dfout


         def prepare_feature(df_feature):
             matrix_feature = (df_feature.copy()).to_numpy()
             matrix_one = np.ones([len(df_feature), 1])
             matrix_feature = np.concatenate((matrix_one, matrix_feature), axis=1)
             return matrix_feature


         def prepare_target(df_target):
```

```
    matrix_target = (df_target.copy()).to_numpy()
    return matrix_target
```

2.7.2 Calculation Functions

```
In [17]: # Calculation Functions
         def calc_linear(X, beta):
             beta_new_rows = int((X.size)/(len(X)))
             beta_new_columns = int((beta.size)/(beta_new_rows))
             beta = beta.reshape(beta_new_rows, beta_new_columns)
             return np.matmul(X, beta)


         def compute_cost(X, y, beta):
             yhat = calc_linear(X, beta)
             yhat_y = yhat - y
             J = (np.matmul((yhat_y).T, yhat_y))/(2*len(X))
             return J


         def gradient_descent(X, y, beta, alpha, num_iters):
             J_storage = np.array([])
             for i in range(num_iters):
                 cost_value = calc_linear(X.T, ((calc_linear(X,beta))-y))/len(X)
                 beta = beta - alpha*cost_value
                 J_storage = np.append(J_storage, cost_value)
             return beta, J_storage


         def predict(df_feature, beta):
             df_feature_z = normalize_z(df_feature.copy())
             X = prepare_feature(df_feature_z)
             yhat = calc_linear(X, beta)
             return yhat


         def linear_regression(X, y, alpha, iterations):
             beta = np.zeros(((X.shape[1]), 1))
             beta, J_storage = gradient_descent(X, y, beta, alpha, iterations)
             yhat = predict(X, beta)
             return beta, J_storage, yhat
```

2.7.3 Metrics Functions

```
In [18]: # Metrics Functions
         def r2_score(y, ypred):
             y_mean = np.mean(y)
             ss_tot = np.sum(np.power((y-y_mean), 2))
```

```
    ss_res = np.sum(np.power(np.subtract(y, ypred), 2))
    return 1 - ((ss_res)/ss_tot)


def mean_squared_error(y, ypred):
    ss_res = np.sum(np.power(np.subtract(y, ypred), 2))
    mse = (ss_res)/(len(y))
    return mse


def adjusted_r2_score(y, ypred, p):
    r2 = r2_score(y,ypred)
    n = y.shape[0]
    return 1 - (((1-r2)*(n-1))/(n-p-1))


def std_dev_score(y, ypred):
    n = y.shape[0]
    return (np.sum(np.subtract(y, ypred))/(n-1))**(0.5)


def std_error_reg_score(y, ypred, p):
    adjusted_r2 = adjusted_r2_score(y, ypred, p)
    std_dev = std_dev_score(y, ypred)
    return ((1-adjusted_r2)**(0.5))*std_dev
```

2.8 Features and Target Dataset Preparation

Describe here what are the features you use and why these features. Put any Python codes to prepare and clean up your features.

Do the same thing for the target. Describe your target and put any codes to prepare your target.

2.8.1 Choice of Features and Target

Features: By understanding the relation between the economic features: 1. GDP per Capita (USD) 2. Annual Inflation Rate (%) 3. Gross per capita Production Index Number (2014-2016 = 100) 4. Binary categorical

we can model them to predict prevalence of undernourishment. We use Binary Categorical as a feature where each country is represented by 0 or 1 (Cambodia:0, Myanmar:1) as a measure to prevent having to make two separate models for the two countries.

Target: We chose prevalence of undernourishment as a (%) as our target, as it gives us information about what percentage of people's food requirements are satisfied, encapsulating the aspect of food security.

```
In [19]: # DESCRIPTION:
         #'Country'
```

```
#'Year'

# TARGET:
#'Percentage of Undernourishment Prevalence (3-Year Average)'

# FEATURES:
#'Binary Categorical'
# 'GDP per Capita (USD)'
# 'Annual Inflation Rate (%)'
# 'Gross per capita Production Index Number (2014-2016 = 100)'
```

2.9 Preparing Training and Test Dataset

```
In [20]: # Extract the features and the target
         features = ['Binary Categorical', 'GDP per Capita (USD)', 'Annual Inflation Rate (%)'
         targets = ['Percentage of Undernourishment Prevalence (3-Year Average)']
         df_features, df_target = get_features_targets(df, features, targets) #DataFrame

         # Split the data set into training and test
         df_features_train, df_features_test, df_target_train, df_target_test = split_data(df_

         # Normalize the features train using z normalization
         df_features_train_z = normalize_z(df_features_train) #DataFrame

         # Prepare the features train and target train to a NumPy
         X = prepare_feature(df_features_train_z) #NumPy 5D
         target = prepare_target(df_target_train) #NumPy 1D
```

```
In [21]: # Display the Descriptive Statistics
         display(df_features.describe())
         display(df_target.describe())
```

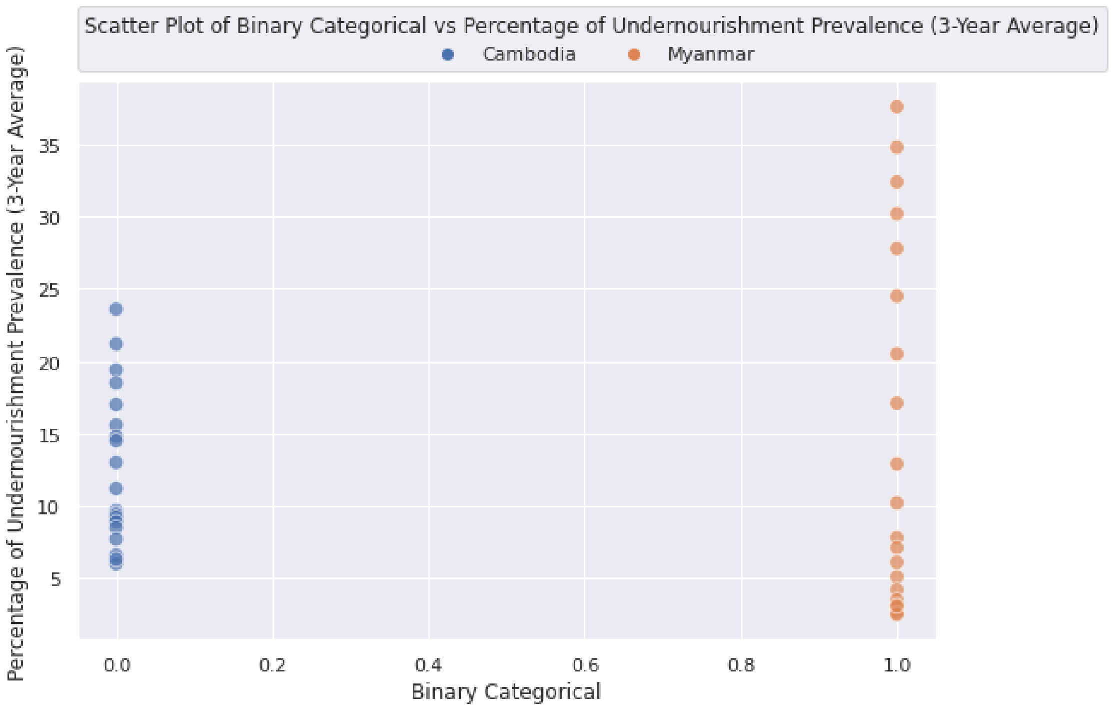
```
       Binary Categorical  GDP per Capita (USD)  Annual Inflation Rate (%)  \
count            40.00000             40.000000                  40.000000
mean              0.50000            825.396363                   9.076909
std               0.50637            454.494599                  11.928724
min               0.00000            128.099702                  -1.241718
25%               0.00000            396.969100                   2.928896
50%               0.50000            833.889141                   4.987380
75%               1.00000           1175.309923                   8.737888
max               1.00000           1643.121389                  57.074511

       Food Production Index (2014-2016 = 100)
count                                40.000000
mean                                 87.907750
std                                  17.631598
min                                  49.460000
25%                                  75.300000
```

```
50%                                   98.010000
75%                                  100.865000
max                                  106.810000


       Percentage of Undernourishment Prevalence (3-Year Average)
count                                            40.000000
mean                                             13.590000
std                                               9.418607
min                                               2.500000
25%                                               6.525000
50%                                               9.950000
75%                                              18.725000
max                                              37.600000
```

2.9.1 Plotting Each Feature with Real Target (All Dataset)

Binary Categorical vs Percentage of Undernourishment Prevalence (3-Year Average)

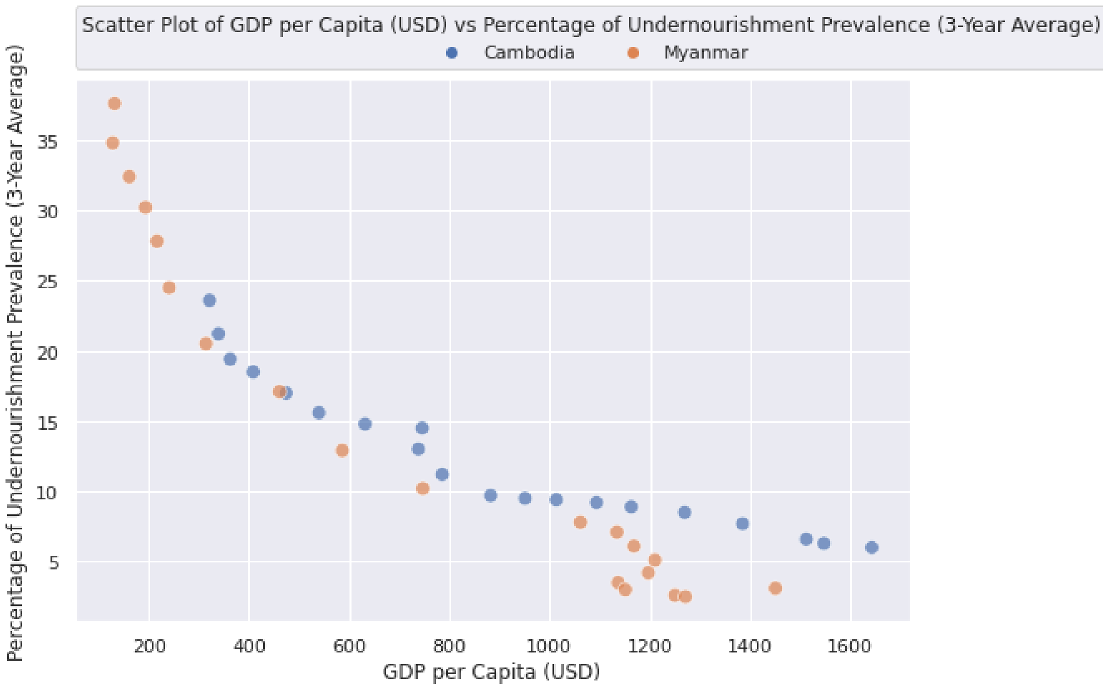
```
In [22]: # Scatter Plot of Binary Categorical vs Percentage of Undernourishment Prevalence (3-
         # ALL DATA SET

         sns.set(rc={'figure.figsize':(9,6)})
         myplot = sns.scatterplot(
                         x = "Binary Categorical", y="Percentage of Undernourishment Prevalence
                         hue="Country", s=75, alpha=0.7)
         myplot.legend(title = "Scatter Plot of " + "Binary Categorical" + " vs " + "Percentage
                 bbox_to_anchor=(0., 1.02, 1., .102), loc=3,
                 ncol=2, borderaxespad=0.)

Out[22]: <matplotlib.legend.Legend at 0x7f2897f27910>
```



GDP per Capita (USD) vs Percentage of Undernourishment Prevalence (3-Year Average)

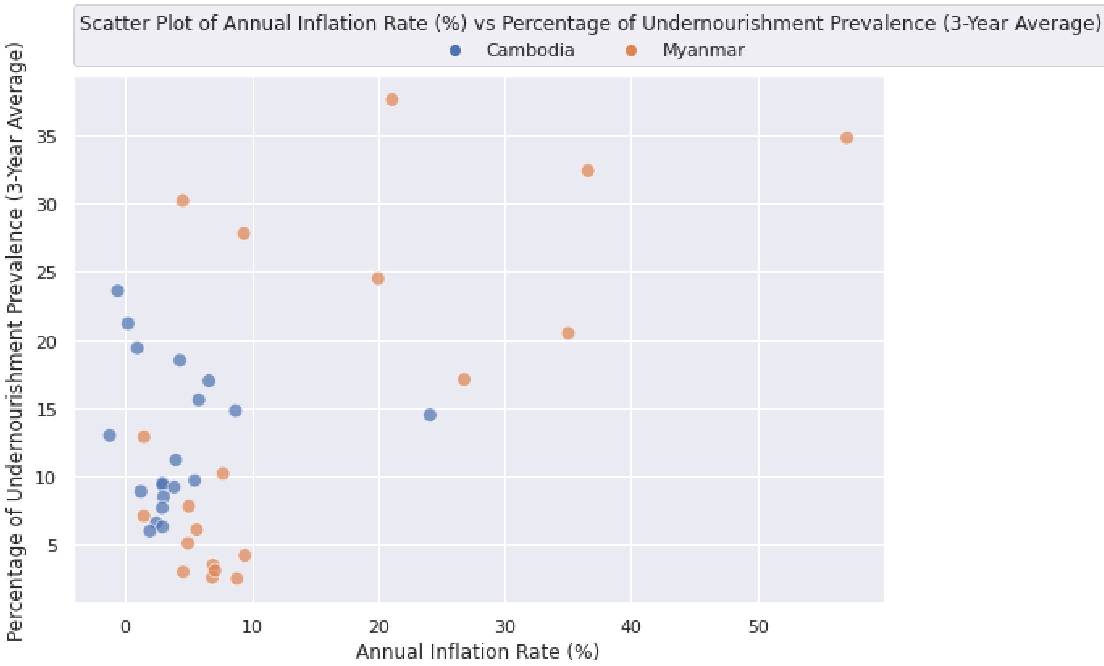
```
In [23]: # Scatter Plot of GDP per Capita (USD) vs Percentage of Undernourishment Prevalence (
         # ALL DATA SET
         sns.set(rc={'figure.figsize':(9,6)})
         myplot = sns.scatterplot(
                         x = "GDP per Capita (USD)", y="Percentage of Undernourishment Prevaler
                         hue="Country", s=75, alpha=0.7)
         myplot.legend(title = "Scatter Plot of " + "GDP per Capita (USD)" + " vs " + "Percenta
                 bbox_to_anchor=(0., 1.02, 1., .102), loc=3,
                 ncol=2, borderaxespad=0.)
```

```
Out[23]: <matplotlib.legend.Legend at 0x7f289766ec90>
```



Annual Inflation Rate (%) vs Percentage of Undernourishment Prevalence (3-Year Average)

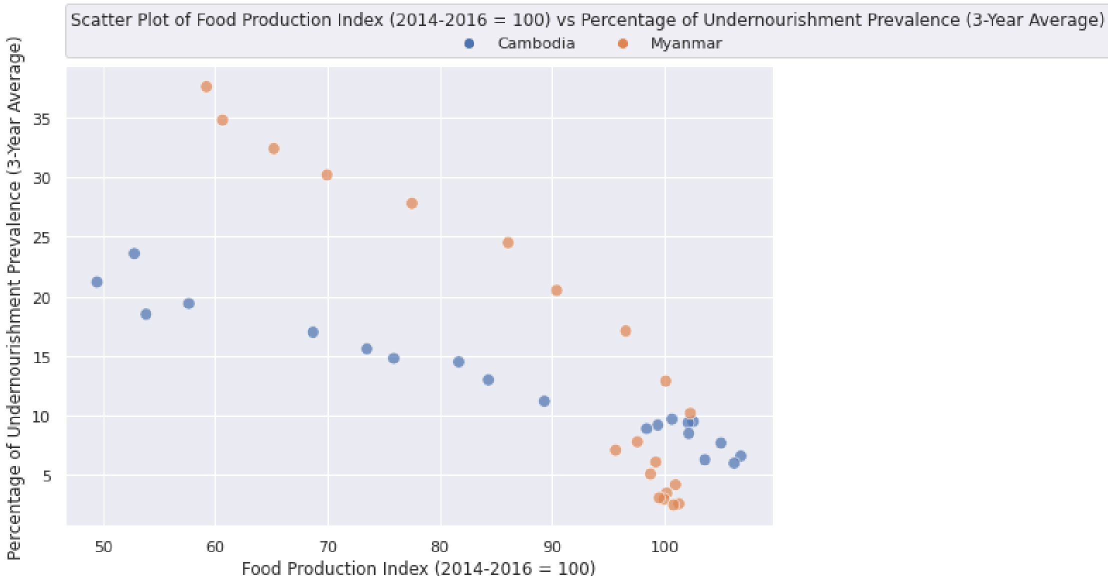
```
In [24]: # Annual Inflation Rate (%) vs Percentage of Undernourishment Prevalence (3-Year Avera
         # ALL DATA SET
         sns.set(rc={'figure.figsize':(9,6)})
         myplot = sns.scatterplot(
                         x = "Annual Inflation Rate (%)", y="Percentage of Undernourishment Pre
                         hue="Country", s=75, alpha=0.7)
         myplot.legend(title = "Scatter Plot of " + "Annual Inflation Rate (%)" + " vs " + "Per
                 bbox_to_anchor=(0., 1.02, 1., .102), loc=3,
                 ncol=2, borderaxespad=0.)

Out[24]: <matplotlib.legend.Legend at 0x7f2895520290>
```



Food Production Index (2014-2016 = 100) vs Percentage of Undernourishment Prevalence (3-Year Average)

In [25]:
```
# Food Production Index (2014-2016 = 100) vs Percentage of Undernourishment Prevalenc
# ALL DATA SET
sns.set(rc={'figure.figsize':(9,6)})
myplot = sns.scatterplot(
                x = "Food Production Index (2014-2016 = 100)", y="Percentage of Underr
                hue="Country", s=75, alpha=0.7)
myplot.legend(title = "Scatter Plot of " + "Food Production Index (2014-2016 = 100)" +
        bbox_to_anchor=(0., 1.02, 1., .102), loc=3,
        ncol=2, borderaxespad=0.)
```

Out[25]:
```
<matplotlib.legend.Legend at 0x7f289543e290>
```



2.10 Building Model

Describe your model. Is this Linear Regression or Logistic Regression? Put any other details about the model. Put the codes to build your model.

2.10.1 Calculating Initial Cost

$J\left(\hat{\beta}_0, \hat{\beta}_1\right) = \frac{1}{2m} \sum_{i=1}^{m} \left(\hat{y}\left(x^i\right) - y^i\right) \times \left(\hat{y}\left(x^i\right) - y^i\right)$

```
In [26]: # Multiple Variables Cost Function
         # Set the value of Beta (same size as features added by column vector of 1)
         beta_multiple = np.zeros(((X.shape[1]), 1)) #NumPy 1D
         J = compute_cost(X, target, beta_multiple)
         print(J)
```

```
[[122.54625]]
```

2.10.2 Model Coefficients and Cost After Multiple Iterations

$\hat{\beta}_0 = \hat{\beta}_0 - \alpha \frac{1}{m} \Sigma_{i=1}^{m} \left(\hat{y}\left(x^i\right) - y^i\right) x_0^i$
$\hat{\beta}_1 = \hat{\beta}_1 - \alpha \frac{1}{m} \Sigma_{i=1}^{m} \left(\hat{y}\left(x^i\right) - y^i\right) x_1^i$
$\hat{\beta}_2 = \hat{\beta}_2 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(\hat{y}\left(x^i\right) - y^i\right) x_2^i$
$\ldots$
$\hat{\beta}_n = \hat{\beta}_n - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(\hat{y}\left(x^i\right) - y^i\right) x_n^i$

```
In [27]: # Beta After Iterations and J After Iterations
         # Set the value of Iterations, Alpha, and Beta
```

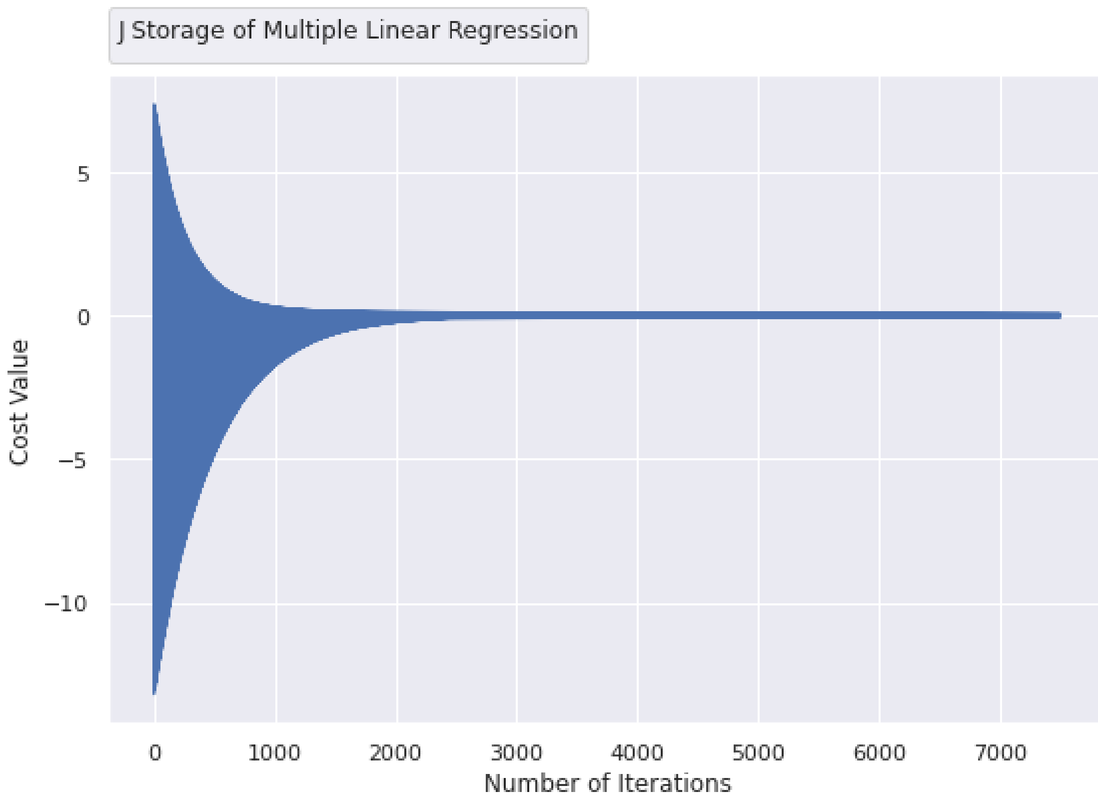
```
alpha = 0.01
iterations = 1500
beta_multiple = np.zeros((((X.shape[1]), 1))

# Call the gradient_descent function
beta_multiple, J_storage_multiple = gradient_descent(X, target, beta_multiple, alpha,
print(beta_multiple)
```

```
[[13.17499626]
 [ 0.35070389]
 [-5.13143412]
 [ 1.16119909]
 [-2.46487031]]
```

In [28]:
```
# Plot the graph of Cost Value in each iteration
sns.set()
myplot = sns.lineplot(x=(np.linspace(start=0, stop=len(J_storage_multiple), num=len(J_
myplot.set_xlabel('Number of Iterations')
myplot.set_ylabel('Cost Value')
myplot.legend(title = "J Storage of Multiple Linear Regression",
        bbox_to_anchor=(0., 1.02, 1., .102), loc=3,
        ncol=2, borderaxespad=0.)
```

```
No handles with labels found to put in legend.
```

Out[28]:
```
<matplotlib.legend.Legend at 0x7f289548c790>
```



2.10.3 Intuitive Analysis of Model Coefficiencts

Intercept = 13.17

GDP per Capita (USD) coefficient: -5.13 (gdp increases, undernourishment decreases) indirectly proportional

Annual Inflation Rate (%) coefficient: 1.16 (inflation increases, undernourishment increases) directly proportional

Gross per capita Production Index Number (2014-2016 = 100) coefficient: -2.46 (agricultural production increases, undernourishment decreases) indirectly proportional

$$\mathbf{X} = \begin{bmatrix} 1 & x_1^1 & \dots & x_n^1 \\ 1 & x_1^2 & \dots & x_n^2 \\ \dots & \dots & \dots & \dots \\ 1 & x_1^m & \dots & x_n^m \end{bmatrix} \in \mathbb{R}^{m\times(n+1)}$$

$$\hat{\mathbf{b}} = \begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \\ \dots \\ \hat{\beta}_n \end{bmatrix} \in \mathbb{R}^{n+1}$$

$$\hat{\mathbf{y}} = \mathbf{X} \times \hat{\mathbf{b}}$$

In [29]:
```
# Predicted Value
# Call the predict() method to get the predicted value of Feature Test
```

```
pred = predict(df_features_test, beta_multiple) #NumPy 1D
display(pred)
```

```
array([[20.79351752],
       [ 6.50075108],
       [10.99367766],
       [26.79523642],
       [12.82301609],
       [ 7.27461179],
       [20.21188228],
       [ 6.3503862 ],
       [29.50615341],
       [ 5.53880542],
       [16.76939911],
       [ 5.2657149 ],
       [ 5.25001583],
       [ 6.12764085],
       [21.90535443],
       [ 8.6937772 ]])
```

2.10.4 Improvement Iterations in Our Model

Prevalence of undernourishment has data from 2000 - 2021. Thus, we had 40 records for our two target countries. Due to our limited dataset, the diversity of training data played a huge role. We made efficient use of the limited data at hand to get the best possible adjusted r2 value and lowest standard error of regression by creating a function to find the best random seed value and therefore, the most diverse training data set. This in turn gave us a good adjusted r2 value and reduced standard error.

We also experimented with different ratios of training and test data sets.

```
In [30]: # ### finding best seed value with test_size=0.4 (After testing, we found that test_s
         # ls_result = []
         # max_result = 0
         # seed_val = 0
         # for val in range(999):
         #     # Split the data set into training and test
         #     df_features_train, df_features_test, df_target_train, df_target_test = split_da

         #     # Normalize the features train using z normalization
         #     df_features_train_z = normalize_z(df_features_train) #DataFrame

         #     # Prepare the features train and target train to a NumPy
         #     X = prepare_feature(df_features_train_z) #NumPy 5D
         #     target = prepare_target(df_target_train) #NumPy 1D
         #     # Beta After Iterations and J After Iterations
         #     # Set the value of Iterations, Alpha
         #     alpha = 0.01
```

```
#     iterations = 1500

#     # Call the gradient_descent function
#     beta_multiple, J_storage_multiple = gradient_descent(X, target, beta_multiple, 
#     adjusted_r2 = adjusted_r2_score(y=prepare_target(df_target_test), ypred=pred, p
#     ls_result.append(adjusted_r2)
#     if max(ls_result)>max_result:
#         max_result = max(ls_result)
#         seed_val = val

# print("max adjusted r-squared:",max_result)
# print("corresponding seed value:", seed_val)

# RESULT
# max adjusted r-squared: 0.8742239115394154
# corresponding seed value: 99
```

2.10.5 Predicting Value of Target Using Trained Model on Test Dataset

In [31]:
```
# Index used in test dataset after splitting data
df_plot_index = list(df_features_test.index)
```

In [32]:
```
# Comparing Real Value of Test Dataset vs Predicted Value of Test Dataset
df_compare_target_predict = df.loc[df_plot_index, ["Country", "Year", "Percentage of U
df_compare_target_predict["Predicted Percentage of Undernourishment Prevalence (3-Yea
df_compare_target_predict = df_compare_target_predict.sort_index(ascending=True)
display(df_compare_target_predict)
df_compare_target_predict = df_compare_target_predict.reset_index(drop=True)
df_compare_target_predict["Features"] = (np.linspace(1, len(df_compare_target_predict

max_val_target_predict = df_compare_target_predict.loc[:, ["Percentage of Undernouris
max_val_target_predict = max(max_val_target_predict.tolist())
```

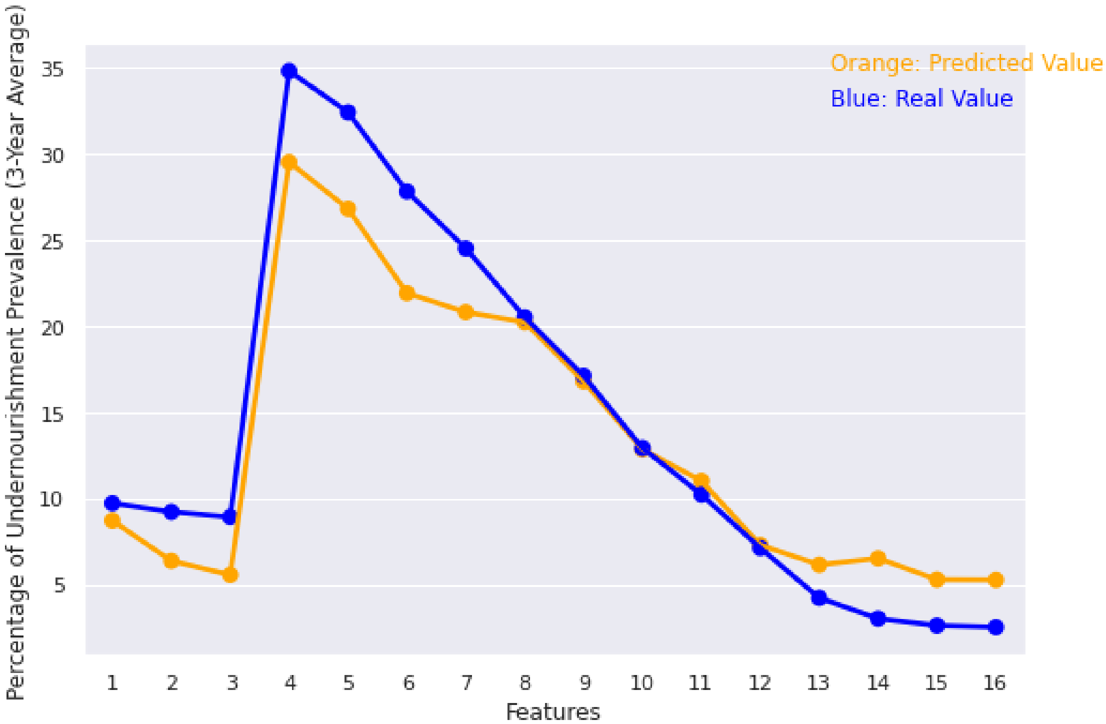
```
     Country  Year  \
10  Cambodia  2011
13  Cambodia  2014
14  Cambodia  2015
21   Myanmar  2002
22   Myanmar  2003
24   Myanmar  2005
25   Myanmar  2006
26   Myanmar  2007
27   Myanmar  2008
28   Myanmar  2009
29   Myanmar  2010
31   Myanmar  2012
34   Myanmar  2015
36   Myanmar  2017
```

```
37   Myanmar  2018
38   Myanmar  2019

    Percentage of Undernourishment Prevalence (3-Year Average)  \
10                                                 9.7
13                                                 9.2
14                                                 8.9
21                                                34.8
22                                                32.4
24                                                27.8
25                                                24.5
26                                                20.5
27                                                17.1
28                                                12.9
29                                                10.2
31                                                 7.1
34                                                 4.2
36                                                 3.0
37                                                 2.6
38                                                 2.5

    Predicted Percentage of Undernourishment Prevalence (3-Year Average)
10                                            8.693777
13                                            6.350386
14                                            5.538805
21                                           29.506153
22                                           26.795236
24                                           21.905354
25                                           20.793518
26                                           20.211882
27                                           16.769399
28                                           12.823016
29                                           10.993678
31                                            7.274612
34                                            6.127641
36                                            6.500751
37                                            5.265715
38                                            5.250016
```

In [33]:
```
sns.set(rc={'figure.figsize':(9,6)})
sns.pointplot(data=df_compare_target_predict,
            x="Features", y="Predicted Percentage of Undernourishment Prevalence (3-Ye
            color="orange", label="Real Value")
#plt.annotate("Real Value", (9.4, 8))
sns.pointplot(data=df_compare_target_predict,
            x="Features", y="Percentage of Undernourishment Prevalence (3-Year Average
            color="blue")
```

```
plt.annotate("Orange: Predicted Value",
             xy=(len(df_compare_target_predict)/1.311475, 1.0027*max_val_target_predic
             color="orange")
plt.annotate("Blue: Real Value",
             xy=(len(df_compare_target_predict)/1.311475, max_val_target_predict/1.06)
             color="blue")
```

Out[33]: Text(12.200003812501192, 32.83018867924528, 'Blue: Real Value')



2.10.6 Combining Dataset for Easier Visualization

In [34]:
```
# Make a new DataFrame for easier visualization

# 1st DataFrame for Real Value
df_plot_1 = df.loc[df_plot_index, :]
real_value_target = df_plot_1.pop("Percentage of Undernourishment Prevalence (3-Year A
df_plot_1["Percentage of Undernourishment Prevalence (3-Year Average)"] = real_value_t
df_plot_1["Value Type"] = ""
df_plot_1.loc[:, "Value Type"] = "Real Value"

# 2nd DataFrame for Predicted Value
df_plot_2 = df.loc[df_plot_index, :]
df_plot_2.pop("Percentage of Undernourishment Prevalence (3-Year Average)")
predicted_value_target = pred
```

```python
    df_plot_2["Percentage of Undernourishment Prevalence (3-Year Average)"] = predicted_va
    df_plot_2["Value Type"] = ""
    df_plot_2.loc[:, "Value Type"] = "Predicted Value"

    # Combine Together 1st DataFrame w/ 2nd DataFrame
    df_plot = pd.DataFrame(pd.concat([df_plot_1, df_plot_2]))
    df_plot = df_plot.sort_index(ascending=True)
    display(df_plot)
    print(df_plot.shape)
```

```
     Country  Year  Binary Categorical  GDP per Capita (USD)  \
10  Cambodia  2011                   0            882.275614
10  Cambodia  2011                   0            882.275614
13  Cambodia  2014                   0           1093.495976
13  Cambodia  2014                   0           1093.495976
14  Cambodia  2015                   0           1162.904995
14  Cambodia  2015                   0           1162.904995
21   Myanmar  2002                   1            128.099702
21   Myanmar  2002                   1            128.099702
22   Myanmar  2003                   1            161.055524
22   Myanmar  2003                   1            161.055524
24   Myanmar  2005                   1            216.311501
24   Myanmar  2005                   1            216.311501
25   Myanmar  2006                   1            240.624014
25   Myanmar  2006                   1            240.624014
26   Myanmar  2007                   1            314.202294
26   Myanmar  2007                   1            314.202294
27   Myanmar  2008                   1            460.908889
27   Myanmar  2008                   1            460.908889
28   Myanmar  2009                   1            586.168180
28   Myanmar  2009                   1            586.168180
29   Myanmar  2010                   1            746.945360
29   Myanmar  2010                   1            746.945360
31   Myanmar  2012                   1           1134.302224
31   Myanmar  2012                   1           1134.302224
34   Myanmar  2015                   1           1196.743333
34   Myanmar  2015                   1           1196.743333
36   Myanmar  2017                   1           1151.114464
36   Myanmar  2017                   1           1151.114464
37   Myanmar  2018                   1           1250.173685
37   Myanmar  2018                   1           1250.173685
38   Myanmar  2019                   1           1271.111536
38   Myanmar  2019                   1           1271.111536

    Annual Inflation Rate (%)  Food Production Index (2014-2016 = 100)  \
10                   5.478447                                   100.67
10                   5.478447                                   100.67
13                   3.855689                                    99.42
```

13	3.855689	99.42
14	1.223932	98.42
14	1.223932	98.42
21	57.074511	60.64
21	57.074511	60.64
22	36.589718	65.22
22	36.589718	65.22
24	9.368618	77.51
24	9.368618	77.51
25	19.996487	86.08
25	19.996487	86.08
26	35.024597	90.42
26	35.024597	90.42
27	26.799537	96.56
27	26.799537	96.56
28	1.472343	100.13
28	1.472343	100.13
29	7.718382	102.32
29	7.718382	102.32
31	1.467583	95.65
31	1.467583	95.65
34	9.454172	101.00
34	9.454172	101.00
36	4.572537	99.96
36	4.572537	99.96
37	6.872329	101.29
37	6.872329	101.29
38	8.825067	100.82
38	8.825067	100.82

	Percentage of Undernourishment Prevalence (3-Year Average) \
10	9.700000
10	8.693777
13	6.350386
13	9.200000
14	5.538805
14	8.900000
21	29.506153
21	34.800000
22	32.400000
22	26.795236
24	21.905354
24	27.800000
25	20.793518
25	24.500000
26	20.211882
26	20.500000
27	16.769399

```
27                                                     17.100000
28                                                     12.900000
28                                                     12.823016
29                                                     10.200000
29                                                     10.993678
31                                                      7.274612
31                                                      7.100000
34                                                      4.200000
34                                                      6.127641
36                                                      6.500751
36                                                      3.000000
37                                                      2.600000
37                                                      5.265715
38                                                      5.250016
38                                                      2.500000

         Value Type
10       Real Value
10  Predicted Value
13  Predicted Value
13       Real Value
14  Predicted Value
14       Real Value
21  Predicted Value
21       Real Value
22       Real Value
22  Predicted Value
24  Predicted Value
24       Real Value
25  Predicted Value
25       Real Value
26  Predicted Value
26       Real Value
27  Predicted Value
27       Real Value
28       Real Value
28  Predicted Value
29       Real Value
29  Predicted Value
31  Predicted Value
31       Real Value
34       Real Value
34  Predicted Value
36  Predicted Value
36       Real Value
37       Real Value
37  Predicted Value
38  Predicted Value
```

```
38        Real Value
```

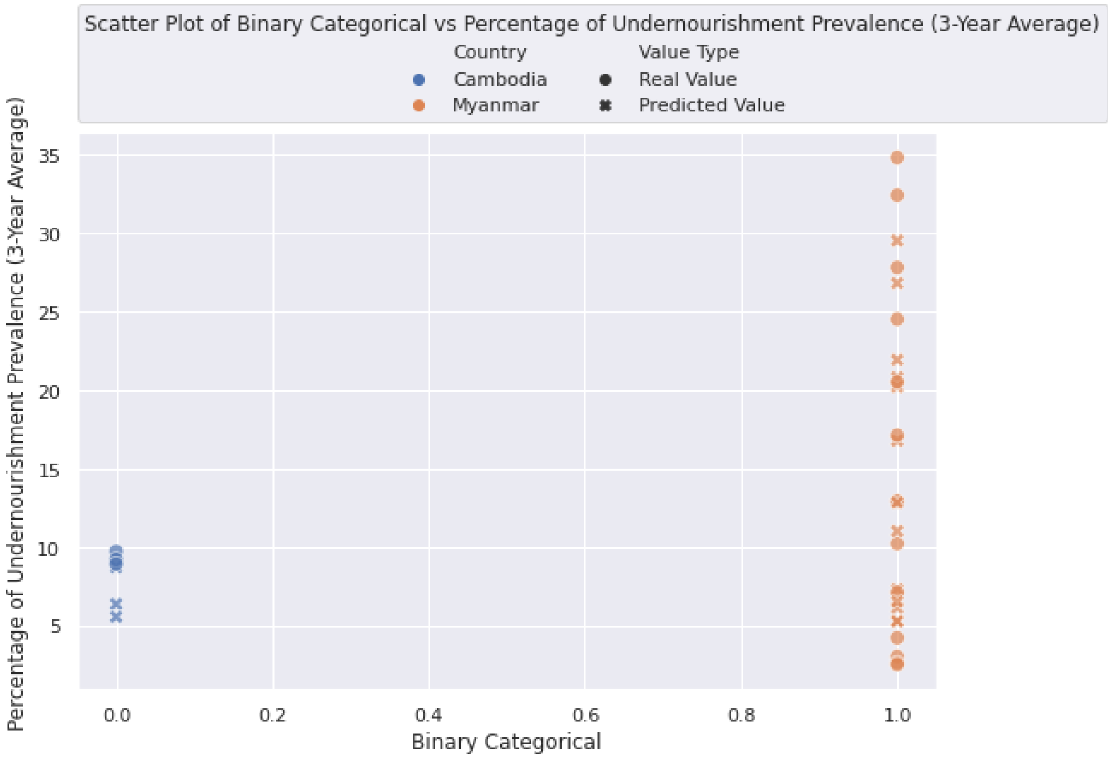
```
(32, 8)
```

2.10.7 Plotting Each Feature with Real Target and Predicted Target (All Dataset)

Binary Categorical vs Percentage of Undernourishment Prevalence (3-Year Average)

```
In [35]: # Scatter Plot of Binary Categorical vs Percentage of Undernourishment Prevalence (3-
         # TARGET DATASET
         sns.set(rc={'figure.figsize':(9,6)})
         myplot = sns.scatterplot(
                         x = "Binary Categorical", y="Percentage of Undernourishment Prevalence
                         style="Value Type", hue="Country", s=75, alpha=0.7)
         myplot.legend(title = "Scatter Plot of " + "Binary Categorical" + " vs " + "Percentage
                 bbox_to_anchor=(0., 1.02, 1., .102), loc=3,
                 ncol=2, borderaxespad=0.)
```

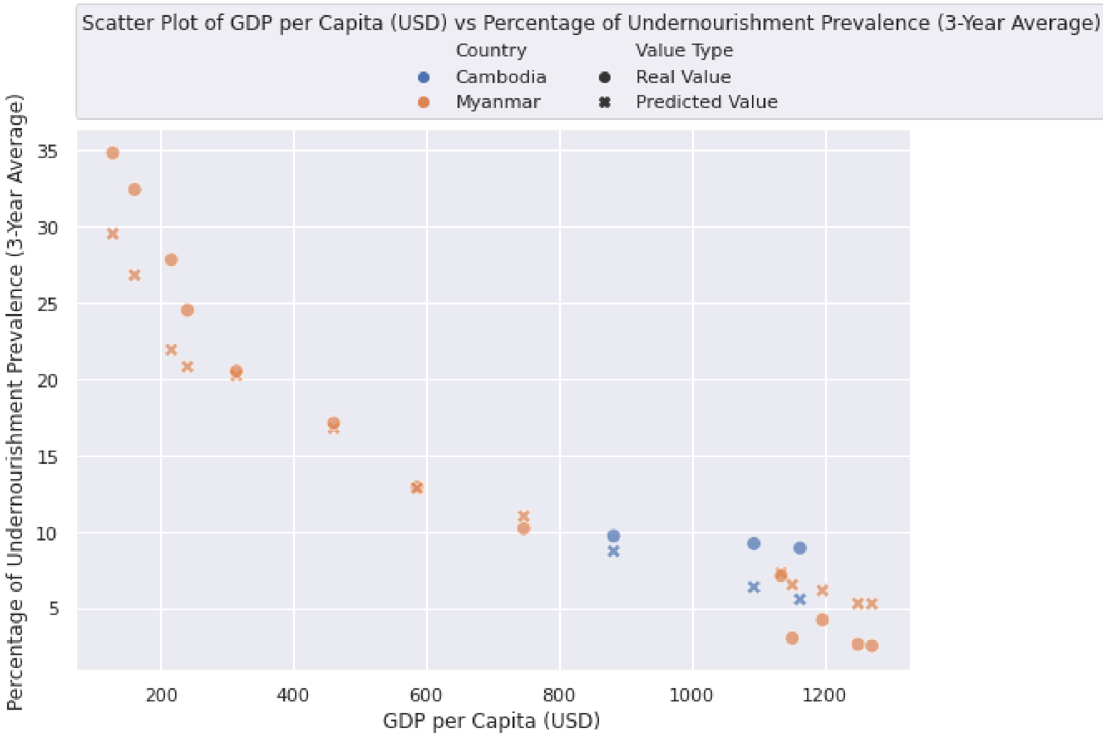
```
Out[35]: <matplotlib.legend.Legend at 0x7f289528ac10>
```



GDP per Capita (USD) vs Percentage of Undernourishment Prevalence (3-Year Average)

```
In [36]: # Scatter Plot of GDP per Capita (USD) vs Percentage of Undernourishment Prevalence (
         # TARGET DATASET
         sns.set(rc={'figure.figsize':(9,6)})
         myplot = sns.scatterplot(
                         x = "GDP per Capita (USD)", y="Percentage of Undernourishment Prevalen
                         style="Value Type", hue="Country", s=75, alpha=0.7)
         myplot.legend(title = "Scatter Plot of " + "GDP per Capita (USD)" + " vs " + "Percenta
                 bbox_to_anchor=(0., 1.02, 1., .102), loc=3,
                 ncol=2, borderaxespad=0.)
```

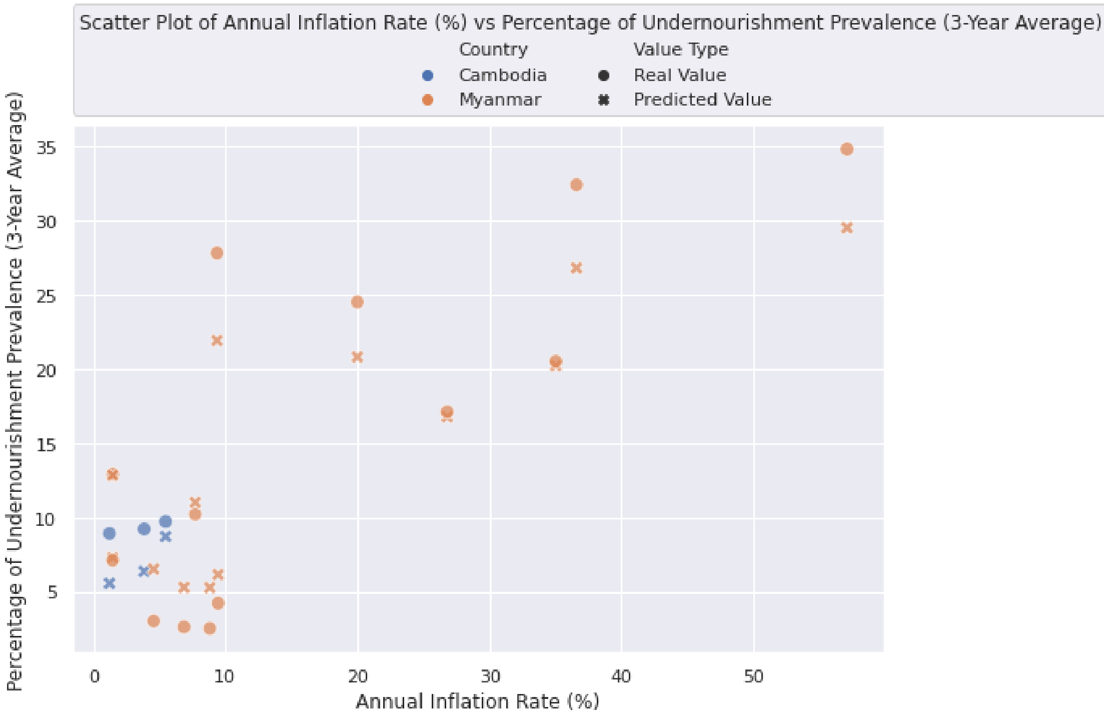
```
Out[36]: <matplotlib.legend.Legend at 0x7f2895035350>
```



Annual Inflation Rate (%) vs Percentage of Undernourishment Prevalence (3-Year Average)

```
In [37]: # Scatter Plot of Annual Inflation Rate (%) vs Percentage of Undernourishment Prevale
         # TARGET DATASET
         sns.set(rc={'figure.figsize':(9,6)})
         myplot = sns.scatterplot(
                         x = "Annual Inflation Rate (%)", y="Percentage of Undernourishment Pre
                         style="Value Type", hue="Country", s=75, alpha=0.7)
         myplot.legend(title = "Scatter Plot of " + "Annual Inflation Rate (%)" + " vs " + "Pe
                 bbox_to_anchor=(0., 1.02, 1., .102), loc=3,
                 ncol=2, borderaxespad=0.)
```

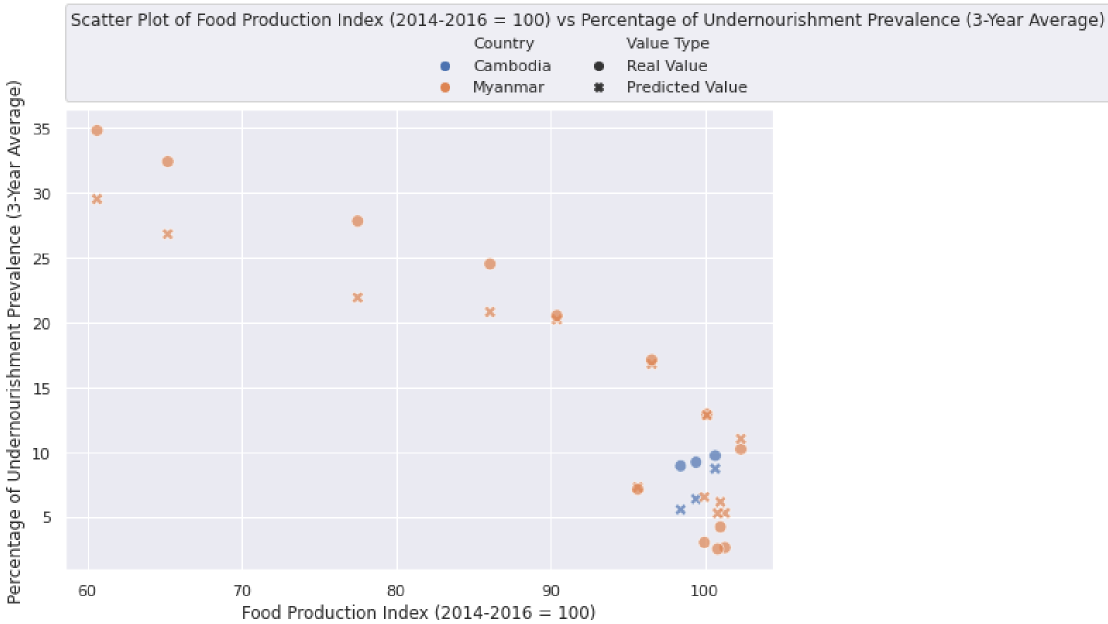
Out[37]: <matplotlib.legend.Legend at 0x7f2895020b90>



Food Production Index (2014-2016 = 100) vs Percentage of Undernourishment Prevalence (3-Year Average)

```
In [38]: # Scatter Plot of Food Production Index (2014-2016 = 100) vs Percentage of Undernouri
         # TARGET DATASET
         sns.set(rc={'figure.figsize':(9,6)})
         myplot = sns.scatterplot(
                         x = "Food Production Index (2014-2016 = 100)", y="Percentage of Under
                         style="Value Type", hue="Country", s=75, alpha=0.7)
         myplot.legend(title = "Scatter Plot of " + "Food Production Index (2014-2016 = 100)" +
                 bbox_to_anchor=(0., 1.02, 1., .102), loc=3,
                 ncol=2, borderaxespad=0.)
```

Out[38]: <matplotlib.legend.Legend at 0x7f2894eace10>



2.11 Evaluating the Model

Describe your metrics and how you want to evaluate your model. Put any Python code to evaluate your model. Use plots to have a visual evaluation.

2.11.1 Standard Metrics

Mean Squared Error (MSE) $\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$

In [39]:
```
# # Mean Squared Error (MSE)
# mse = mean_squared_error(y=prepare_target(df_target_test), ypred=pred)
# print(mse)
# Mean Squared Error (MSE)
mse = mean_squared_error(prepare_target(df_target_test), pred)
print(mse)
```

```
9.990736660164742
```

Coefficient of Determination (Rš) $R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$

In [40]:
```
# Coefficient of Determination (Rš)
# Calculate r2 score by calling a function, the arguments must be a NumPy
r2 = r2_score(prepare_target(df_target_test), pred)
print(r2)
```

```
0.9077642017955713
```

2.11.2 Relevant Metrics

Adjusted Coefficient of Determination (Adjusted Rš) We cannot use Rš to evaluate multiple linear regression due to the following issues 1. Every time we add a predictor to a model, the R-squared increases, even if due to chance alone. It never decreases. Consequently, a model with more terms may appear to have a better fit simply because it has more terms. 2. If a model has too many predictors and higher order polynomials, it begins to model the random noise in the data. This leads to overfitting the model and it produces misleadingly high R-squared values and a lessened ability to make predictions.

Adjusted R-squared is a modified version of R-squared that has been adjusted for the number of predictors in the model. Adjusted R-squared helps us determine how much of the correlation with the index is due to the addition of those variables. The adjusted R-squared compensates for the addition of variables and only increases if the new predictor enhances the model. (Side note: The Adjusted R2 Score is an extension of the R2 Score that takes into account sample size and the number of independent variables. Hence, it is possible for it to return zero or negative values in the event of a lack of data when doing regression)

$R^2_{adj} = 1 - \frac{(1-R^2)(n-1)}{n-p-1}$ $n: number of dataset$ $p: number of features$

```
In [41]: # Adjusted Coefficient of Determination (Adjusted Rš)
         # Calculate adjusted r2 score by calling a function, the arguments must be a NumPy fo
         # p is the number of Independent Variables, which are the length of Features list
         adjusted_r2 = adjusted_r2_score(y=prepare_target(df_target_test), ypred=pred, p=len(fe
         print(adjusted_r2)
```

```
0.8742239115394154
```

Standard Error of Regression (S) In addition to the adjusted R-squared metric, we also used standard error or regression.

The standard error of the regression (S), represents the average distance that the observed values fall from the regression line. S is in the units of the dependent variable.

It is particularly useful because we can use it to assess the precision of predictions. Roughly 95% of the observation should fall within +/- two standard errors of the regression, which is a quick approximation of a 95% prediction interval.

$S = \sqrt{1 - R^2_{adj}} \cdot S.D$

$S.D = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}$

```
In [42]: # Standard Error of Regression (S)
         # p is the number of Independent Variables, which are the length of Features list
         std_eror_reg = std_error_reg_score(y=prepare_target(df_target_test), ypred=pred, p=len
         print(std_eror_reg)
```

```
0.3730853869367281
```

2.12 Verifying Process with Scikit Sklearn

```
In [43]: from sklearn.linear_model import LinearRegression
         from sklearn.model_selection import train_test_split
         from sklearn.metrics import r2_score, mean_squared_error
```

```
In [44]: # Extract the features and the target
         df_features_scikit, df_target_scikit = get_features_targets(df, ['Binary Categorical'

         # Normalize the features train using z normalization
         df_features_scikit = normalize_z(df_features_scikit)

         # Split the data into training and test data set using scikit-learn function
         df_features_train_scikit, df_features_test_scikit, df_target_train_scikit, df_target_

         # Instantiate LinearRegression() object
         model = LinearRegression()

         # Call the fit() method and find the Beta Value
         model.fit(df_features_train_scikit, df_target_train_scikit)
         print(model.coef_, model.intercept_)

         # Call the predict() method
         pred_scikit = model.fit(df_features_train_scikit, df_target_train_scikit).predict(df_
```

```
[[ 0.32477307 -5.66203628  2.25446523 -1.60300413]] [14.3862165]
```

```
In [45]: # Call the r2_score method and find the r2 value
         print(r2_score(df_target_test_scikit, pred_scikit))
```

```
0.8759086808324694
```

2.13 Improving the Model

Discuss any steps you can do to improve the models. Put any python codes. You can repeat the steps above with the codes to show the improvement in the accuracy.

- Adding more dataset to improve precision of the model
  1. More countries
  2. More years
- Adding more features that is relevant to improve accuracy of the model
- Adding more targets to better analyse and predict food security and food safety
- Adding interactive window with input() function to allow one to choose which features and targets one wants to model and select what types of graphs one wishes to see
- Using more Python library such as TensorFlow to get the derivatives of the Cost Function, instead of using Gradient Descent

2.14 Discussion and Analysis

Discuss your model and accuracy in solving the problem. Analyze the results of your metrics. Put any conclusion here.

Our model has pretty good accuracy in predicting Percentage of Prevalence of Undernourishment with adjusted r-squared value = 0.8742239115394154 and Standard error of regression (S) = 0.3730853869367281. Percentage of Prevalence of Undernourishment gives us information about the food security of the country. With this data we can predict food security of the country in the future and take necessary steps and prepare in advance to avoid low levels of food security.

```
In [ ]:
```