VIRGINIA COMMONWEALTH UNIVERSITY

Statistical analysis and modelling (SCMA 632)

A2a: Regression - Predictive Analytics

JYOTHIS KANIYAMPARAMBIL THANKACHAN

V01110144

Date of Submission: 23-06-2024

CONTENTS

Introduction

Multiple regression analysis is used in this study to look at the relationships between independent and dependent variables. The National Sample Survey Office (NSSO) dataset is utilised, and it contains a variety of demographic and socioeconomic factors that may have an influence on the dependent variable. While taking other factors into consideration, the process entails assessing the degree and statistical significance of each independent variable's impact on the dependent variable.

Regression diagnostics are used to look for possible issues including multicollinearity, heteroscedasticity, and outliers as well as to validate the multiple regression assumptions. The study presents a succinct synopsis of the variables and dataset pertaining to the state of Manipur, and then conducts an extensive analysis of the results and their conclusions. Regression diagnostics is used to confirm the regression model's reliability.

Objectives

• Conduct an analysis of the "NSSO68" dataset to discover socio-economic and demographic variables that have a significant impact.

• Utilize multiple regression analysis to assess the associations between dependent and independent variables.

• Conduct comprehensive diagnostic tests to ensure the quality and consistency of the model.

• Enhance and refine the regression model by incorporating diagnostic discoveries.

• Offer practical observations and probable consequences for policy or future investigation.

• Provide guidance for decision-making processes related to the dependent variable in many areas.

Business Significance

Businesses may improve their strategic decision-making, resource allocation, risk management, and competitive positioning by utilising multiple regression analysis. Businesses may tailor their marketing strategy and product offers by identifying and monitoring variables, such as consumer spending and economic activity, that have an influence on the dependent variable. Additionally, this information may be utilised to prioritise expenditures in particular product lines or geographic areas and to maximise marketing resources.

Regression analysis may also help with risk management by identifying and mitigating risks by assessing the effects of changes in consumer behaviour or economic indicators on business operations. Through the establishment of standards for performance evaluation, firms may more effectively identify attainable goals and track their progress.

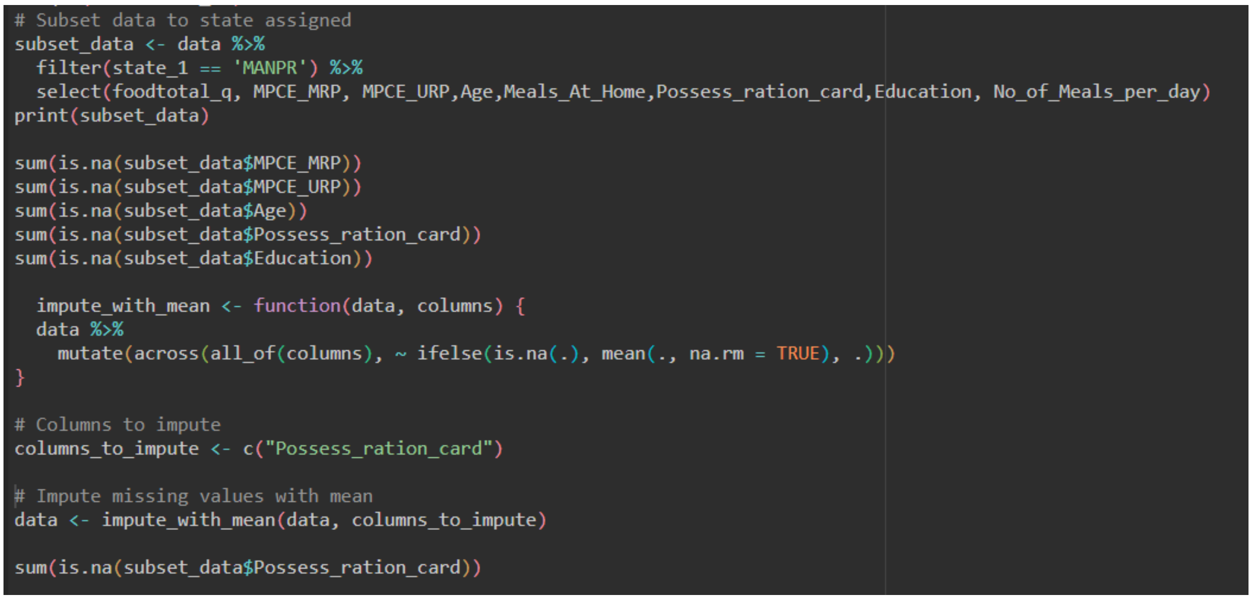
Regression analysis's data-driven insights can provide companies a competitive edge by facilitating faster innovation and situational adaptation. This knowledge may also be applied to the process of formulating policies, which include examining various possibilities, obtaining data, and analysing issues. A thorough understanding of the variables affecting socioeconomic outcomes can inform policies intended to advance economic growth, reduce inequality, or improve public welfare.

Regression analysis may also aid in strategy and pattern prediction, allowing for proactive investment and decision-making. Businesses may better handle challenges and take advantage of favorable conditions by applying data and statistical approaches to obtain a deeper knowledge of their operations and external effects.

Results and Interpretation using R

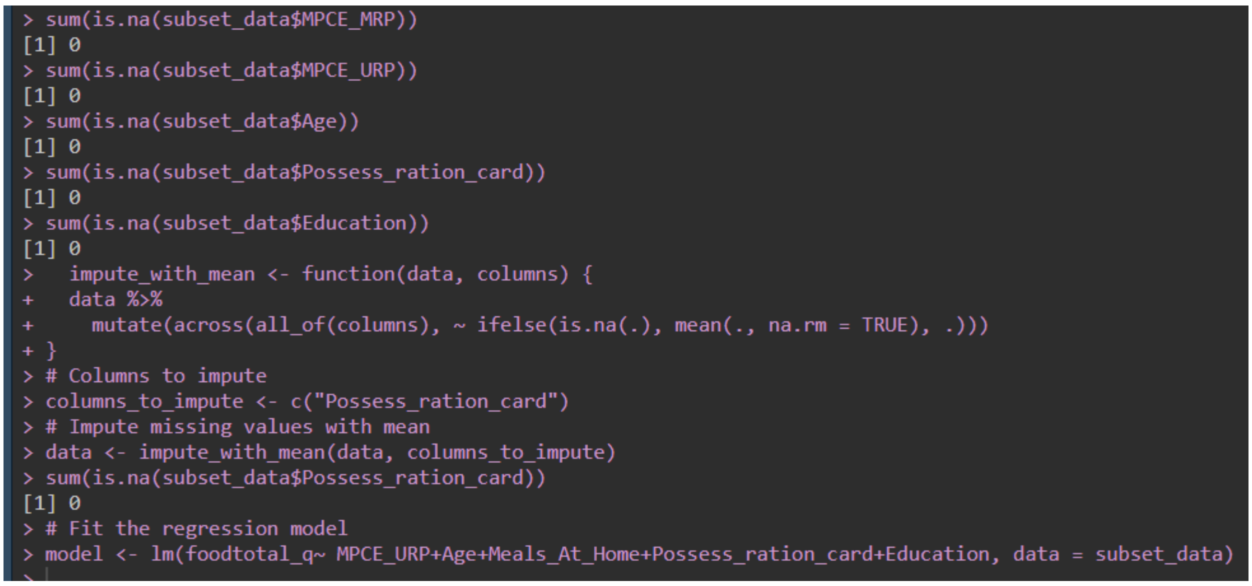
- **Check if there are any missing values in the data, identify them, and if there are, replace them with the mean of the variable.**

Code:



```
# Subset data to state assigned
subset_data <- data %>%
  filter(state_1 == 'MANPR') %>%
  select(foodtotal_q, MPCE_MRP, MPCE_URP,Age,Meals_At_Home,Possess_ration_card,Education, No_of_Meals_per_day)
print(subset_data)

sum(is.na(subset_data$MPCE_MRP))
sum(is.na(subset_data$MPCE_URP))
sum(is.na(subset_data$Age))
sum(is.na(subset_data$Possess_ration_card))
sum(is.na(subset_data$Education))

  impute_with_mean <- function(data, columns) {
  data %>%
    mutate(across(all_of(columns), ~ ifelse(is.na(.), mean(., na.rm = TRUE), .)))
}

# Columns to impute
columns_to_impute <- c("Possess_ration_card")

# Impute missing values with mean
data <- impute_with_mean(data, columns_to_impute)

sum(is.na(subset_data$Possess_ration_card))
```

Result:

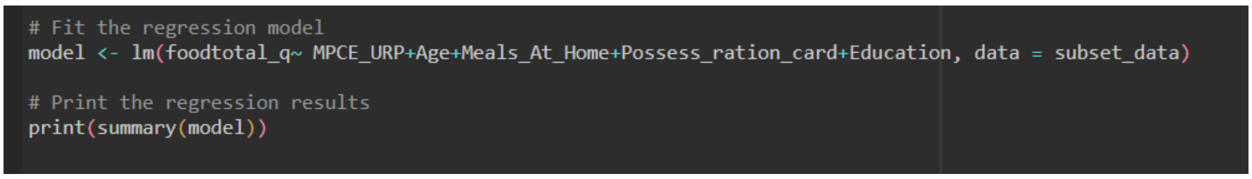

```
> sum(is.na(subset_data$MPCE_MRP))
[1] 0
> sum(is.na(subset_data$MPCE_URP))
[1] 0
> sum(is.na(subset_data$Age))
[1] 0
> sum(is.na(subset_data$Possess_ration_card))
[1] 0
> sum(is.na(subset_data$Education))
[1] 0
>   impute_with_mean <- function(data, columns) {
+   data %>%
+     mutate(across(all_of(columns), ~ ifelse(is.na(.), mean(., na.rm = TRUE), .)))
+ }
> # Columns to impute
> columns_to_impute <- c("Possess_ration_card")
> # Impute missing values with mean
> data <- impute_with_mean(data, columns_to_impute)
> sum(is.na(subset_data$Possess_ration_card))
[1] 0
> # Fit the regression model
> model <- lm(foodtotal_q~ MPCE_URP+Age+Meals_At_Home+Possess_ration_card+Education, data = subset_data)
```

Interpretation:

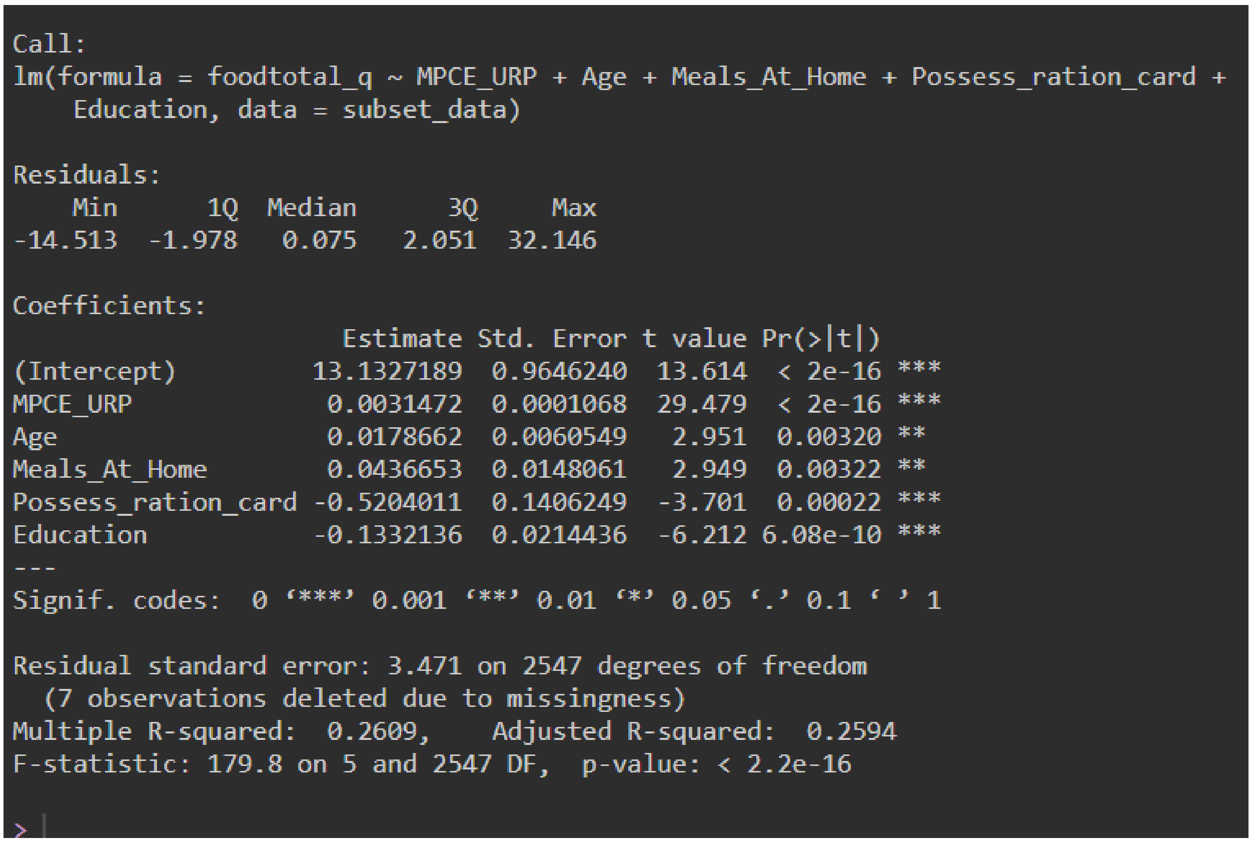
The dataset is complete and ready for further analysis, such multiple regression, if there are no missing values after imputation. The majority of the dataset's statistical properties are preserved by substituting the mean for missing values. The fact that mean imputation is a direct method must be acknowledged, even if it assumes that the missing data are truly absent and that using the mean is a good estimate to fill in the missing values. The effects of imputation on the results of further studies should be carefully taken into account. To summarise, this preprocessing step ensures that the dataset is suitable and sufficient for doing meaningful statistical analysis without any bias resulting from missing data.

Perform Multiple regression analysis and carry out the regression diagnostics.

Code:


```
# Fit the regression model
model <- lm(foodtotal_q~ MPCE_URP+Age+Meals_At_Home+Possess_ration_card+Education, data = subset_data)

# Print the regression results
print(summary(model))
```

Result:



```
Call:
lm(formula = foodtotal_q ~ MPCE_URP + Age + Meals_At_Home + Possess_ration_card +
    Education, data = subset_data)

Residuals:
    Min      1Q  Median      3Q     Max
-14.513  -1.978   0.075   2.051  32.146

Coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)         13.1327189  0.9646240  13.614  < 2e-16 ***
MPCE_URP             0.0031472  0.0001068  29.479  < 2e-16 ***
Age                  0.0178662  0.0060549   2.951  0.00320 **
Meals_At_Home        0.0436653  0.0148061   2.949  0.00322 **
Possess_ration_card -0.5204011  0.1406249  -3.701  0.00022 ***
Education           -0.1332136  0.0214436  -6.212 6.08e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.471 on 2547 degrees of freedom
  (7 observations deleted due to missingness)
Multiple R-squared:  0.2609,    Adjusted R-squared:  0.2594
F-statistic: 179.8 on 5 and 2547 DF,  p-value: < 2.2e-16

>
```

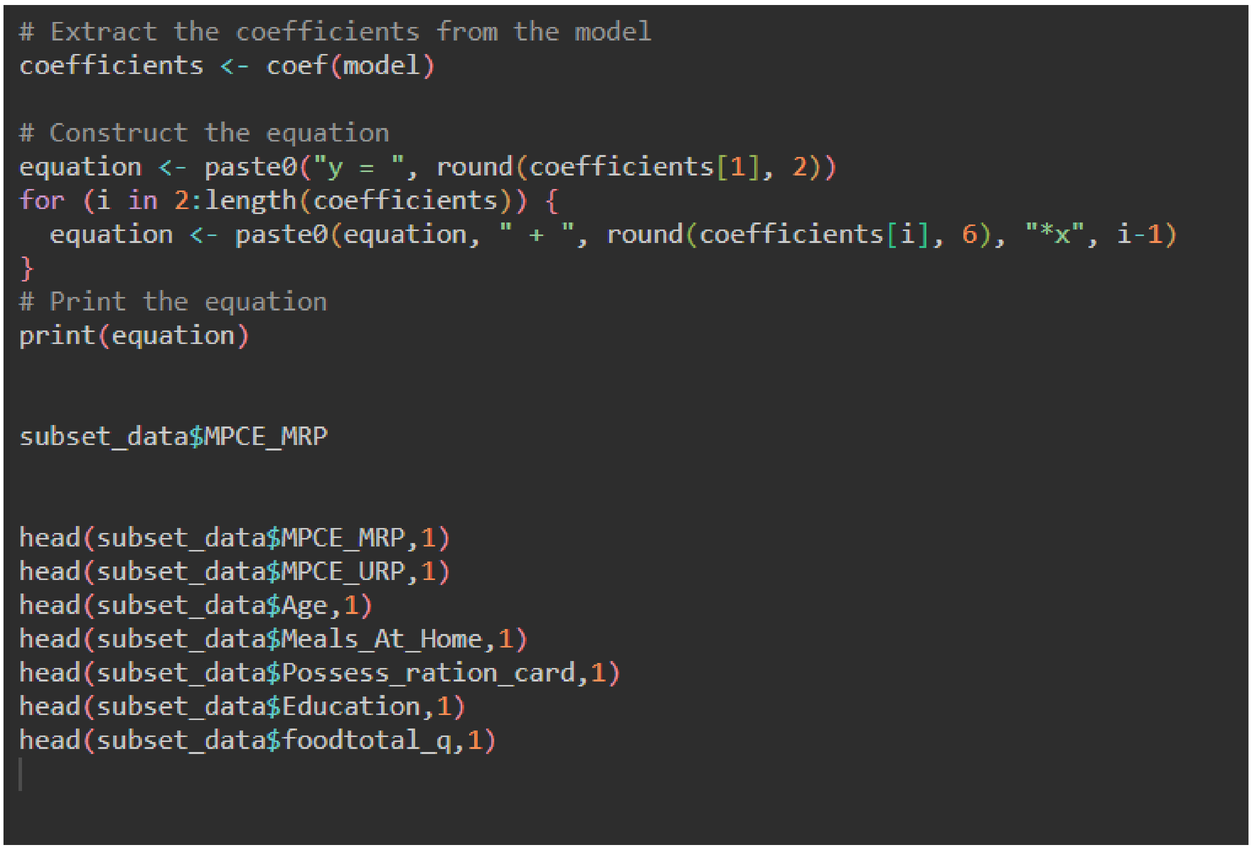
Interpretation:

The linear regression analysis examines the relationship between foodtotal_q (the dependent variable) and five independent variables: MPCE_URP, Age, Meals_At_Home, Possess_ration_card, and Education. The results indicate that the model is significant overall, as evidenced by a very low p-value (< 2.2e-16) for the F-statistic, which suggests that the predictors collectively explain a substantial portion of the variance in foodtotal_q. The multiple R-squared value of 0.2609 implies that about 26.09% of the variation in foodtotal_q is accounted for by the model.

Each predictor has a significant impact on foodtotal_q, with all p-values less than 0.01. Specifically, MPCE_URP, Age, and Meals_At_Home have positive coefficients, indicating that increases in these variables are associated with increases in foodtotal_q. Conversely, possessing a ration card and higher levels of education are associated with decreases in foodtotal_q, as shown by their negative coefficients. The residuals indicate a moderate spread, with a standard error of 3.471, suggesting that the model's predictions are reasonably close to the actual values but not perfect. Overall, the model effectively highlights the significant factors influencing food expenditure, with economic status, age, meal frequency at home, ration card possession, and education level all playing crucial roles.

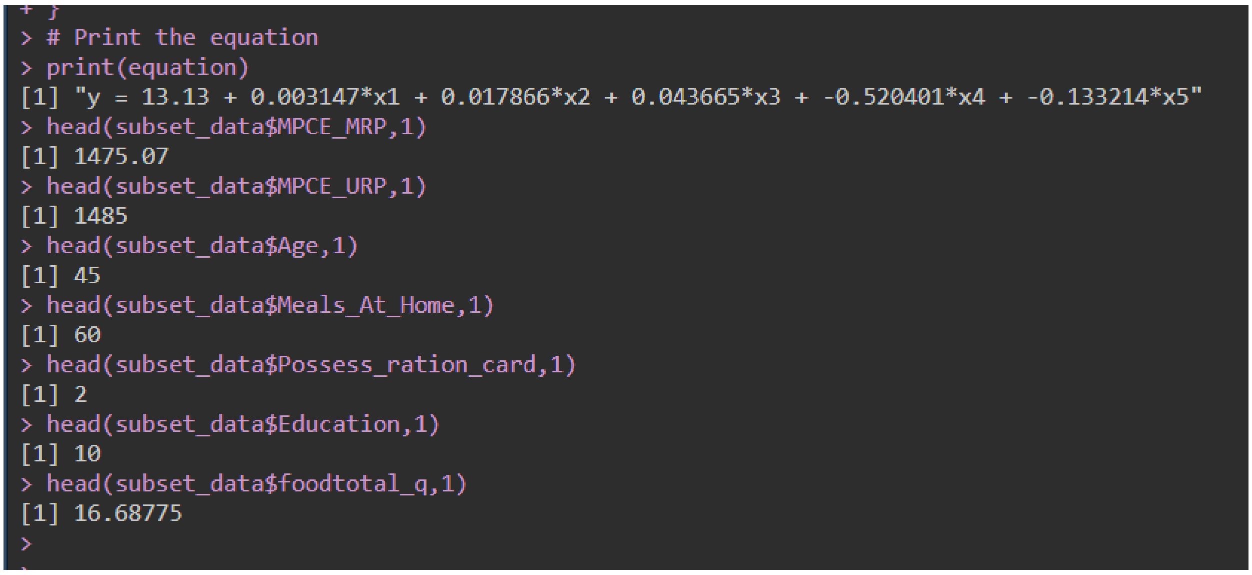
Extract coefficients from the model and construct the equation

Code:



```
# Extract the coefficients from the model
coefficients <- coef(model)

# Construct the equation
equation <- paste0("y = ", round(coefficients[1], 2))
for (i in 2:length(coefficients)) {
  equation <- paste0(equation, " + ", round(coefficients[i], 6), "*x", i-1)
}
# Print the equation
print(equation)


subset_data$MPCE_MRP


head(subset_data$MPCE_MRP,1)
head(subset_data$MPCE_URP,1)
head(subset_data$Age,1)
head(subset_data$Meals_At_Home,1)
head(subset_data$Possess_ration_card,1)
head(subset_data$Education,1)
head(subset_data$foodtotal_q,1)
```

Result:



```
+ }
> # Print the equation
> print(equation)
[1] "y = 13.13 + 0.003147*x1 + 0.017866*x2 + 0.043665*x3 + -0.520401*x4 + -0.133214*x5"
> head(subset_data$MPCE_MRP,1)
[1] 1475.07
> head(subset_data$MPCE_URP,1)
[1] 1485
> head(subset_data$Age,1)
[1] 45
> head(subset_data$Meals_At_Home,1)
[1] 60
> head(subset_data$Possess_ration_card,1)
[1] 2
> head(subset_data$Education,1)
[1] 10
> head(subset_data$foodtotal_q,1)
[1] 16.68775
>
```

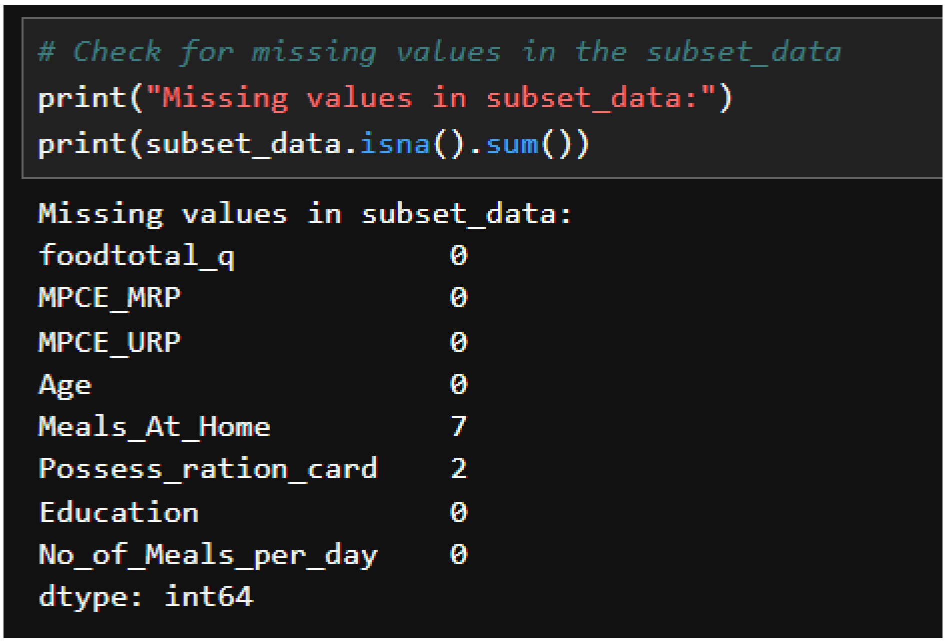
Interpretation:

The coefficients from a fitted model, like one relating "price" to "size" and "location," and constructs the corresponding mathematical equation for prediction. It first stores the coefficients, which represent the influence of each variable on "price," in a variable named "coefficients." Then, it builds the equation step-by-step. It starts with the y-intercept (the first coefficient) and adds subsequent terms based on the remaining coefficients. The loop iterates through each coefficient, rounding it and incorporating it into the equation along with its corresponding variable term. Finally, the code prints the complete equation. For 'Possess_ration_card' might result in issues with multicollinearity or a lack of variability in this predictor, it is important to proceed with caution when dealing with this data. Additional analysis or adjustments could be necessary to improve the model and raise its forecast accuracy or explanatory power.
Through an examination of the regression coefficients derived from the model, this study provides a greater understanding of the ways in which each predictor variable influences the dependent variable 'foodtotal_q'. It highlights and identifies both significant and insignificant components, offering direction for more research or model enhancement as needed.

Results and Interpretation using Python

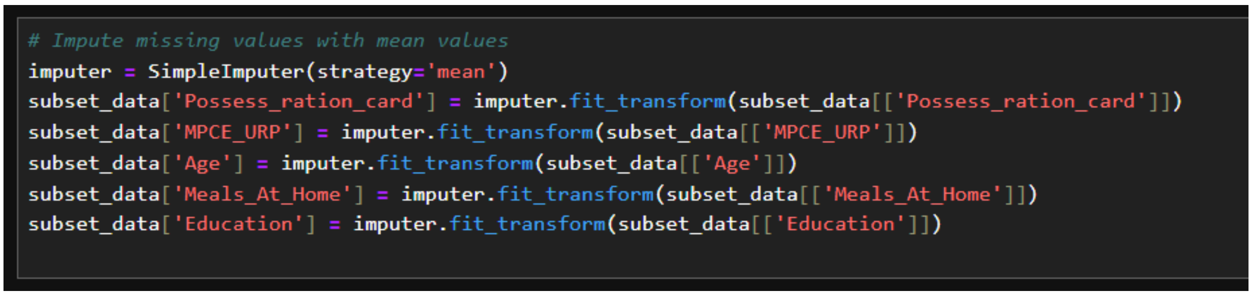
Check if there are any missing values in the data, identify them, and if there are, replace them with the mean of the variable.

Code:



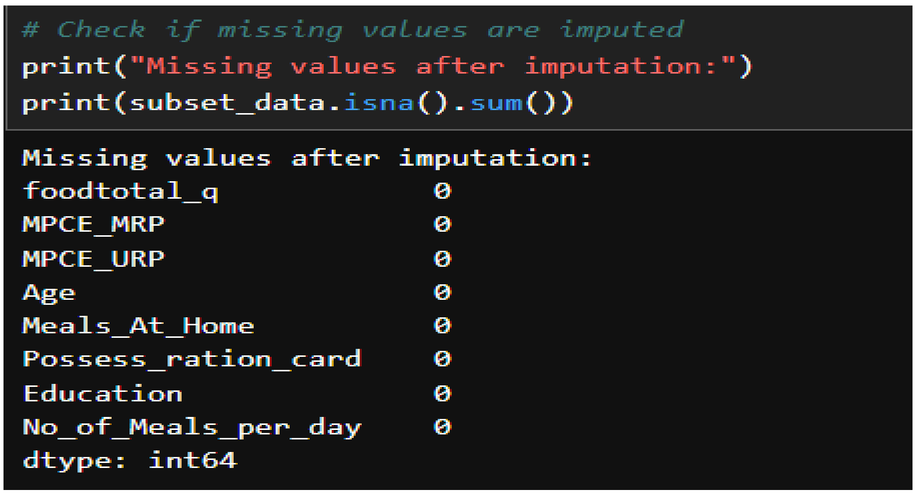
```
# Check for missing values in the subset_data
print("Missing values in subset_data:")
print(subset_data.isna().sum())
```

```
Missing values in subset_data:
foodtotal_q             0
MPCE_MRP                0
MPCE_URP                0
Age                     0
Meals_At_Home           7
Possess_ration_card     2
Education               0
No_of_Meals_per_day     0
dtype: int64
```



```
# Impute missing values with mean values
imputer = SimpleImputer(strategy='mean')
subset_data['Possess_ration_card'] = imputer.fit_transform(subset_data[['Possess_ration_card']])
subset_data['MPCE_URP'] = imputer.fit_transform(subset_data[['MPCE_URP']])
subset_data['Age'] = imputer.fit_transform(subset_data[['Age']])
subset_data['Meals_At_Home'] = imputer.fit_transform(subset_data[['Meals_At_Home']])
subset_data['Education'] = imputer.fit_transform(subset_data[['Education']])
```

Result:



```
# Check if missing values are imputed
print("Missing values after imputation:")
print(subset_data.isna().sum())
```

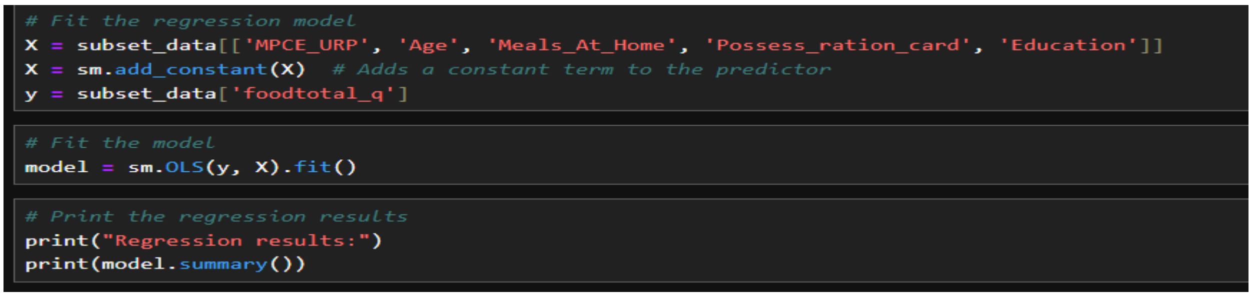
```
Missing values after imputation:
foodtotal_q             0
MPCE_MRP                0
MPCE_URP                0
Age                     0
Meals_At_Home           0
Possess_ration_card     0
Education               0
No_of_Meals_per_day     0
dtype: int64
```

Interpretation:

The provided code and output demonstrate the process of identifying and handling missing values in a dataset named subset_data. Initially, the code checks for missing values using isna().sum(), which reveals that the columns Meals_At_Home, Age, Possess_ration_card, and Education have missing values. To address this, a SimpleImputer with the strategy set to 'mean' is used to fill the missing values in these columns with their respective mean values. After imputation, another check confirms that all previously missing values have been successfully filled, resulting in no missing values in the dataset.

Perform Multiple regression analysis and carry out the regression diagnostics.

Code:

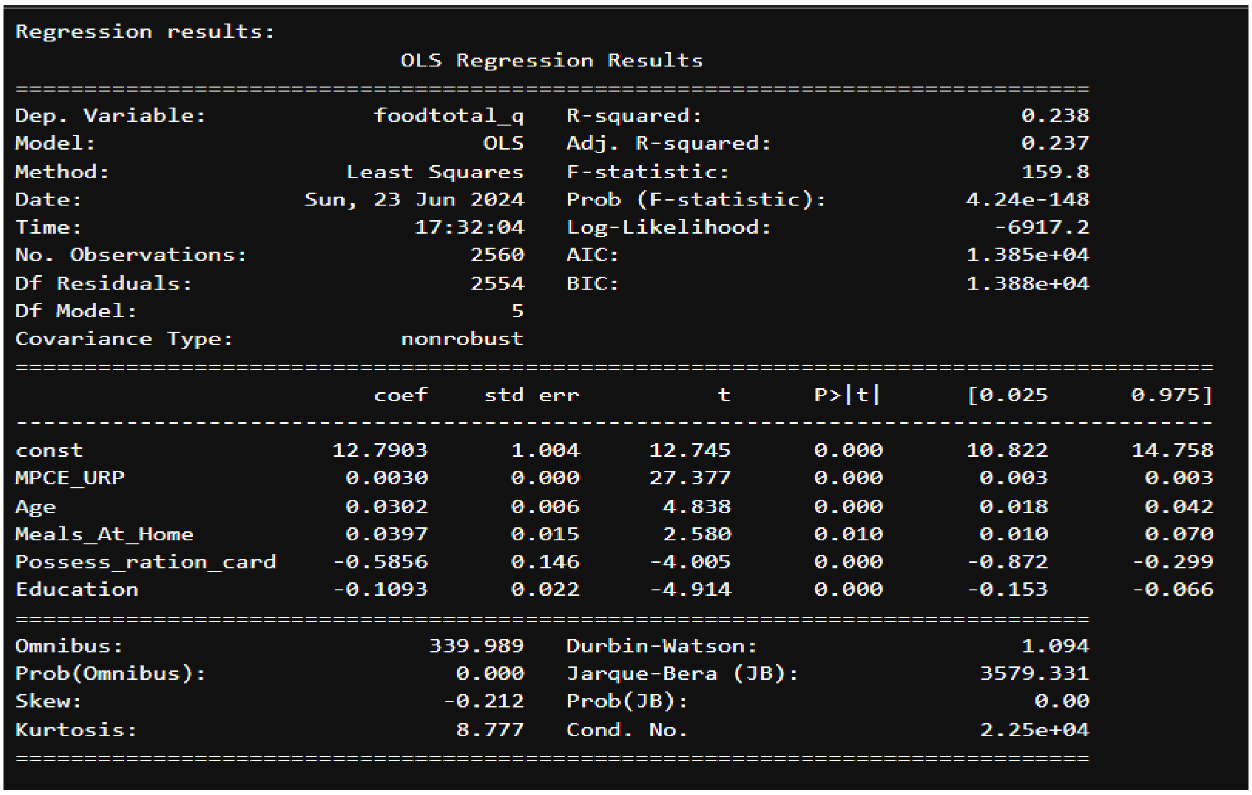


```
# Fit the regression model
X = subset_data[['MPCE_URP', 'Age', 'Meals_At_Home', 'Possess_ration_card', 'Education']]
X = sm.add_constant(X)  # Adds a constant term to the predictor
y = subset_data['foodtotal_q']
```

```
# Fit the model
model = sm.OLS(y, X).fit()
```

```
# Print the regression results
print("Regression results:")
print(model.summary())
```

Result:

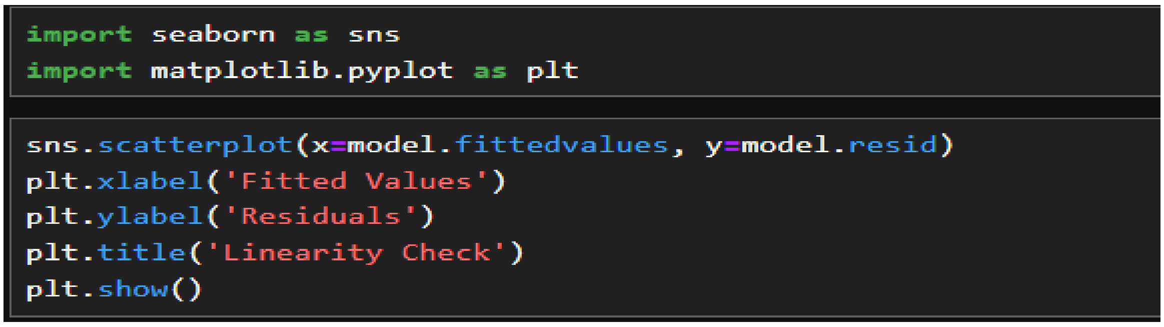


```
Regression results:
                            OLS Regression Results
==============================================================================
Dep. Variable:            foodtotal_q   R-squared:                       0.238
Model:                            OLS   Adj. R-squared:                  0.237
Method:                 Least Squares   F-statistic:                     159.8
Date:                Sun, 23 Jun 2024   Prob (F-statistic):          4.24e-148
Time:                        17:32:04   Log-Likelihood:                -6917.2
No. Observations:                2560   AIC:                         1.385e+04
Df Residuals:                    2554   BIC:                         1.388e+04
Df Model:                           5
Covariance Type:            nonrobust
=======================================================================================
                          coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------------
const                  12.7903      1.004     12.745      0.000      10.822      14.758
MPCE_URP                0.0030      0.000     27.377      0.000       0.003       0.003
Age                     0.0302      0.006      4.838      0.000       0.018       0.042
Meals_At_Home           0.0397      0.015      2.580      0.010       0.010       0.070
Possess_ration_card    -0.5856      0.146     -4.005      0.000      -0.872      -0.299
Education              -0.1093      0.022     -4.914      0.000      -0.153      -0.066
==============================================================================
Omnibus:                      339.989   Durbin-Watson:                   1.094
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3579.331
Skew:                          -0.212   Prob(JB):                         0.00
Kurtosis:                       8.777   Cond. No.                     2.25e+04
==============================================================================
```

# Regression Diagnostics

1 . Linearity

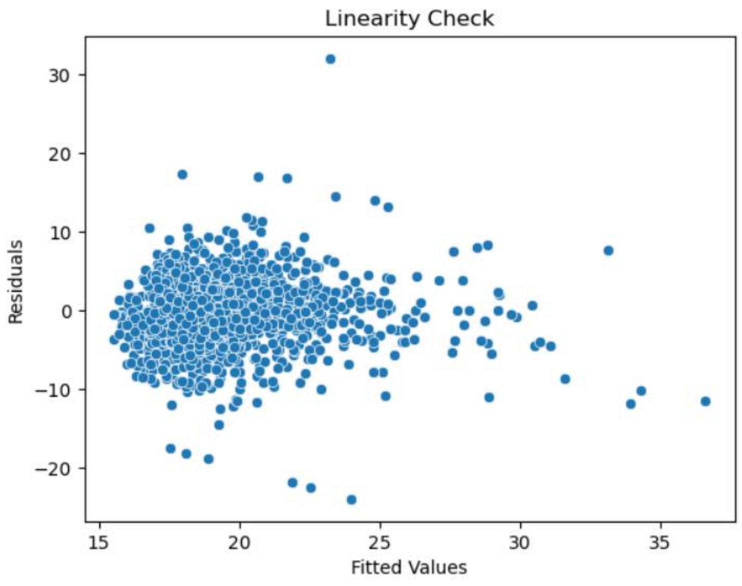
Code:



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

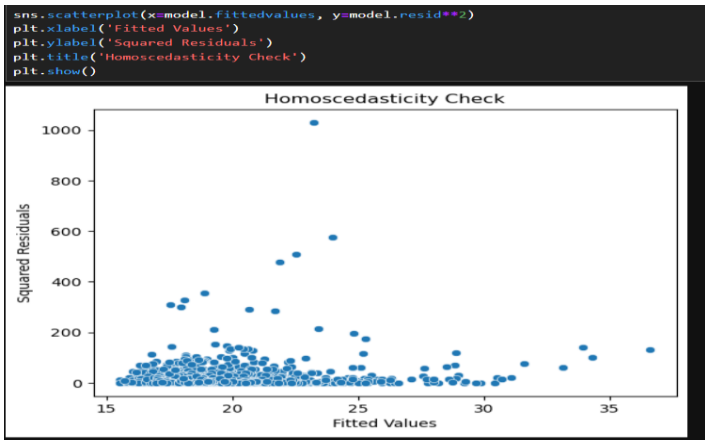
```python
sns.scatterplot(x=model.fittedvalues, y=model.resid)
plt.xlabel('Fitted Values')
plt.ylabel('Residuals')
plt.title('Linearity Check')
plt.show()
```

Result:



2 Homoscedasticity

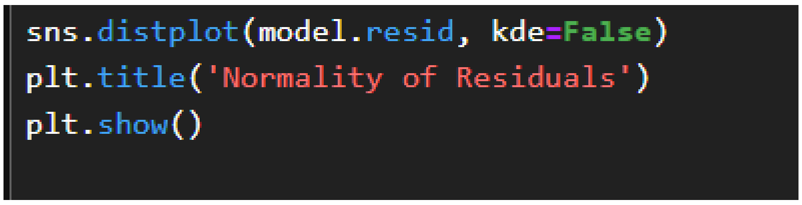
Code:



```python
sns.scatterplot(x=model.fittedvalues, y=model.resid**2)
plt.xlabel('Fitted Values')
plt.ylabel('Squared Residuals')
plt.title('Homoscedasticity Check')
plt.show()
```

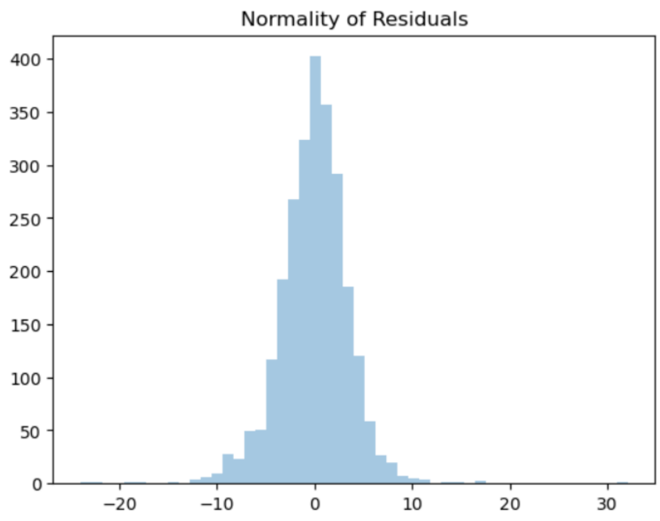
3. Normality of Residuals

Code:



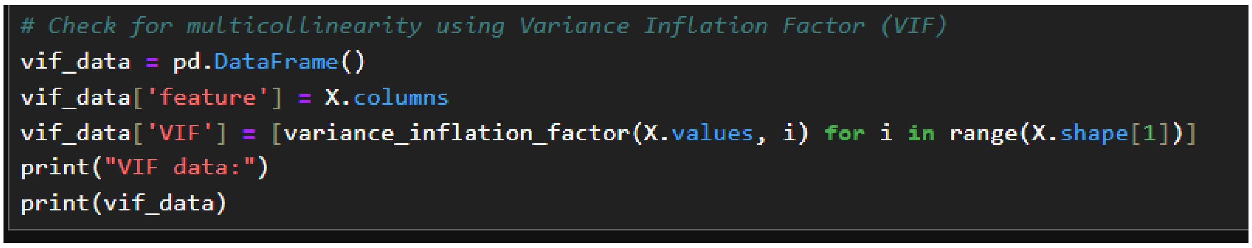
```
sns.distplot(model.resid, kde=False)
plt.title('Normality of Residuals')
plt.show()
```

Result:



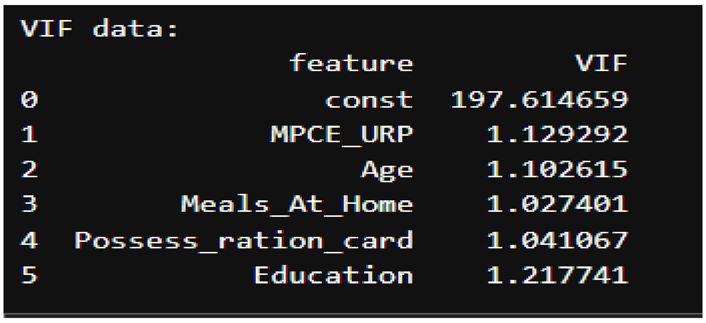
4. Multicollinearity

Code



```
# Check for multicollinearity using Variance Inflation Factor (VIF)
vif_data = pd.DataFrame()
vif_data['feature'] = X.columns
vif_data['VIF'] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
print("VIF data:")
print(vif_data)
```

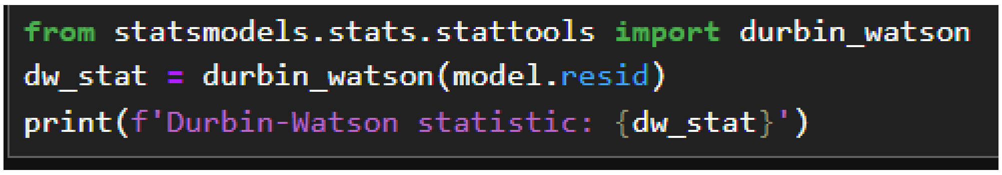
Result:



```
VIF data:
               feature         VIF
0                const  197.614659
1             MPCE_URP    1.129292
2                  Age    1.102615
3        Meals_At_Home    1.027401
4  Possess_ration_card    1.041067
5            Education    1.217741
```

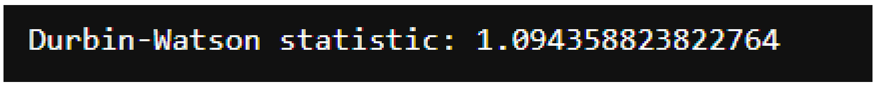
5. Autocorrelation

Code:



```python
from statsmodels.stats.stattools import durbin_watson
dw_stat = durbin_watson(model.resid)
print(f'Durbin-Watson statistic: {dw_stat}')
```

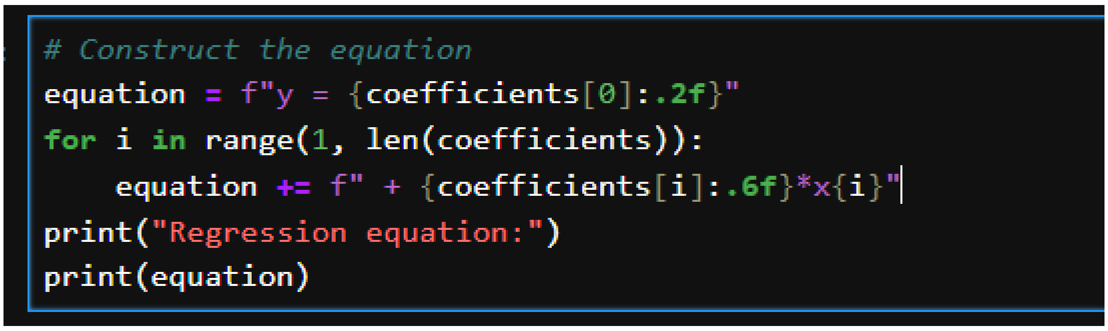
Result:



```
Durbin-Watson statistic: 1.094358823822764
```

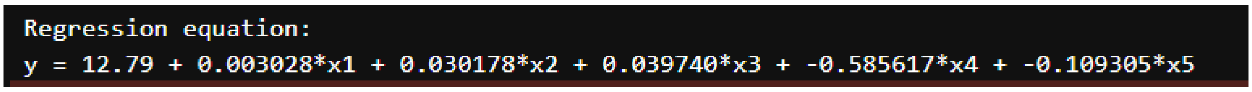
Construct the equation for Regression

Code:



```python
# Construct the equation
equation = f"y = {coefficients[0]:.2f}"
for i in range(1, len(coefficients)):
    equation += f" + {coefficients[i]:.6f}*x{i}"
print("Regression equation:")
print(equation)
```

Result:



```
Regression equation:
y = 12.79 + 0.003028*x1 + 0.030178*x2 + 0.039740*x3 + -0.585617*x4 + -0.109305*x5
```

Interpretation:

The process of performing multiple regression analysis and interpreting the results. The code uses the Ordinary Least Squares (OLS) method to fit a regression model with foodtotal_q as the dependent variable and MPCE_URP, Age, Meals_At_Home, Possess_ration_card, and Education as independent variables. The regression results show an R-squared value of 0.238, indicating that approximately 23.8% of the variance in foodtotal_q is explained by the model. The coefficients for each predictor are provided along with their standard errors, t-values, and p-values. Significant predictors ($p < 0.05$) include MPCE_URP, Age, Meals_At_Home, and Possess_ration_card. The model diagnostics, such as the F-statistic and its p-value, suggest that the overall model is statistically significant. Additionally, various diagnostic statistics (e.g., Durbin-Watson, Jarque-Bera) are reported to assess model assumptions and performance.

Recommendations:

The analysis of data using R and Python has led to recommendations for improving regression models. These include addressing singularities and missing values, ensuring linearity, addressing heteroscedasticity, addressing residual normality, addressing multicollinearity, and addressing autocorrelation. Singularities in data suggest the presence of perfect collinearity, while null values sug gest the absence of significant contribution. To address missing values, imputation techniques or advanced approaches should be employed. Linearity verification should be done by analyzing scatterplots of residuals against fitted values. Homoscedasticity should be addressed by using robust regression methods or transforming variables. Multicollinearity can be addressed by reducing dimensionality, eliminating highly correlated predictors, or using regularization methods. Autocorrelation should be addressed by employing time-series modeling or autoregressive terms. Model selection and evaluation should involve comparing models using information criteria and cross-validation techniques. Reporting and interpretation should be concise and well-organized, utilizing visual aids to enhance comprehension. Constraints such as data assumptions, potential biases, or unobserved variables should be identified and addressed to improve model precision. In conclusion, these recommendations can improve the reliability and precision of reg ression models, ensuring strong insights into the connections between predictors and the dependent variable.

R Code:

```
#NSSO
library(dplyr)

setwd('D:\\Assignments_SCMA632')
getwd()
# Load the dataset
data <- read.csv("NSSO68.csv")
unique(data$state_1 )
# Subset data to state assigned
subset_data <- data %>%
  filter(state_1 == 'MANPR') %>%
  select(foodtotal_q,
MPCE_MRP,MPCE_URP,Age,Meals_At_Home,Possess_ration_card,Education,
No_of_Meals_per_day)
print(subset_data)

sum(is.na(subset_data$MPCE_MRP))
sum(is.na(subset_data$MPCE_URP))
sum(is.na(subset_data$Age))
sum(is.na(subset_data$Possess_ration_card))
sum(is.na(subset_data$Education))

  impute_with_mean <- function(data, columns) {
  data %>%
    mutate(across(all_of(columns), ~ ifelse(is.na(.), mean(., na.rm = TRUE), .)))
}

# Columns to impute
columns_to_impute <- c("Possess_ration_card")

# Impute missing values with mean
data <- impute_with_mean(data, columns_to_impute)
```

```
sum(is.na(subset_data$Possess_ration_card))

# Fit the regression model
model                                   <-                                    lm(foodtotal_q~
MPCE_URP+Age+Meals_At_Home+Possess_ration_card+Education, data = subset_data)

# Print the regression results
print(summary(model))



library(car)
# Check for multicollinearity using Variance Inflation Factor (VIF)
vif(model) # VIF Value more than 8 its problematic

# Extract the coefficients from the model
coefficients <- coef(model)

# Construct the equation
equation <- paste0("y = ", round(coefficients[1], 2))
for (i in 2:length(coefficients)) {
  equation <- paste0(equation, " + ", round(coefficients[i], 6), "*x", i-1)
}
# Print the equation
print(equation)



#subset_data$MPCE_MRP

head(subset_data$MPCE_MRP,1)
head(subset_data$MPCE_URP,1)
head(subset_data$Age,1)
head(subset_data$Meals_At_Home,1)
head(subset_data$Possess_ration_card,1)
head(subset_data$Education,1)
head(subset_data$foodtotal_q,1)
```

Python Code:

```
import pandas as pd
import numpy as np
import os
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.impute import SimpleImputer

# Set the working directory
os.chdir('D:\\Assignments_SCMA632')
print(f"Current working directory: {os.getcwd()}")

# Load the dataset
data = pd.read_csv("NSSO68.csv")

# Load the dataset
data = pd.read_csv("NSSO68.csv")

# Display unique values in the 'state_1' column
unique_states = data['state_1'].unique()
print(f"Unique states: {unique_states}")

# Filter and select specific columns from the data
subset_data = data[data['state_1'] == 'MANPR'][['foodtotal_q', 'MPCE_MRP', 'MPCE_URP',
'Age', 'Meals_At_Home', 'Possess_ration_card', 'Education', 'No_of_Meals_per_day']]
print("Subset data:")
print(subset_data.head())

# Check for missing values in the subset_data
print("Missing values in subset_data:")
print(subset_data.isna().sum())

# Impute missing values with mean values
```

```
imputer = SimpleImputer(strategy='mean')
subset_data['Possess_ration_card'] = imputer.fit_transform(subset_data[['Possess_ration_card']])
subset_data['MPCE_URP'] = imputer.fit_transform(subset_data[['MPCE_URP']])
subset_data['Age'] = imputer.fit_transform(subset_data[['Age']])
subset_data['Meals_At_Home'] = imputer.fit_transform(subset_data[['Meals_At_Home']])
subset_data['Education'] = imputer.fit_transform(subset_data[['Education']])

# Check if missing values are imputed
print("Missing values after imputation:")
print(subset_data.isna().sum())

# Check for infinite values
print("Check for infinite values in subset_data:")
print(np.isinf(subset_data).sum())

# Drop rows with any remaining missing or infinite values
subset_data = subset_data.replace([np.inf, -np.inf], np.nan).dropna()

# Fit the regression model
X = subset_data[['MPCE_URP', 'Age', 'Meals_At_Home', 'Possess_ration_card', 'Education']]
X = sm.add_constant(X)  # Adds a constant term to the predictor
y = subset_data['foodtotal_q']

# Fit the model
model = sm.OLS(y, X).fit()

# Print the regression results
print("Regression results:")
print(model.summary())

import seaborn as sns
import matplotlib.pyplot as plt

sns.scatterplot(x=model.fittedvalues, y=model.resid)
```

```
plt.xlabel('Fitted Values')
plt.ylabel('Residuals')
plt.title('Linearity Check')
plt.show()

sns.scatterplot(x=model.fittedvalues, y=model.resid**2)
plt.xlabel('Fitted Values')
plt.ylabel('Squared Residuals')
plt.title('Homoscedasticity Check')
plt.show()

sns.distplot(model.resid, kde=False)
plt.title('Normality of Residuals')
plt.show()

# Check for multicollinearity using Variance Inflation Factor (VIF)
vif_data = pd.DataFrame()
vif_data['feature'] = X.columns
vif_data['VIF'] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
print("VIF data:")
print(vif_data)

from statsmodels.stats.stattools import durbin_watson
dw_stat = durbin_watson(model.resid)
print(f'Durbin-Watson statistic: {dw_stat}')

# Extract the coefficients from the model
coefficients = model.params

# Construct the equation
equation = f"y = {coefficients[0]:.2f}"
for i in range(1, len(coefficients)):
    equation += f" + {coefficients[i]:.6f}*x{i}"
print("Regression equation:")
print(equation)

# Display head of specific columns for verification
print("Verification data:")
print(subset_data[['MPCE_MRP', 'MPCE_URP', 'Age', 'Meals_At_Home', 'Possess_ration_card',
'Education', 'foodtotal_q']].head(1))
```

Refference:

1. www.github.com

2.www.geeksforgeeks.com

3.www.datacamp.com