Data Visualization Analysis

Hello, Learner! I see you've uploaded a dataset named `cacao.csv`. Let's start by loading and examining the data to understand its structure. Once we have a good grasp of the dataset, you can go ahead and ask any questions or give instructions based on your needs.
Let's load and take a look at the first few rows of your dataset.
I'll start by loading the data.

```python
import pandas as pd

# Load the dataset
file_path = '/mnt/data/cacao.csv'
cacao_df = pd.read_csv(file_path)

# Display the first few rows of the dataset
cacao_df.head()
```

```
Ergebnis
   ant_exclusion  stem_diameter  height    canopy  dw_healthy  dw_infect  \
0              0        22.1500  247.50  0.473333        1222        159
1              1        23.9500  223.75  0.603333        1850        216
2              0        23.2500  266.50  0.556667        1058         58
3              1        21.8500  234.25  0.298333         635         52
4              0        26.4875  285.25  0.483333        1145         72

   dw_total  fw_pulb  fw_seeds  fw_total  ab_fl_op  ab_fl_cl  ab_fl
0      1382    13788      5314     19103      2018      5602   7720
1      2076    20222      8417     28639      1257      3648   4894
2      1117     9121      3044     12105      1523      3970   5484
3       688     9236      3739     12975      1312      3484   4795
4      1218    13417      3911     17328      2620      5792   8409
```

Great! We have successfully loaded your dataset. Here are the first few rows to give us an idea of the data we are working with:
| | ant_exclusion | stem_d

Sure, let's start with the histograms.

Histograms

1. **Code for the histograms and execution:**

```python
import matplotlib.pyplot as plt

# List of columns to plot histograms for
columns_to_plot = ['stem_diameter', 'height', 'canopy', 'dw_healthy', 'dw_infect',
'dw_total', 'fw_pulb', 'fw_seeds', 'fw_total', 'ab_fl_op', 'ab_fl_cl', 'ab_fl']
```

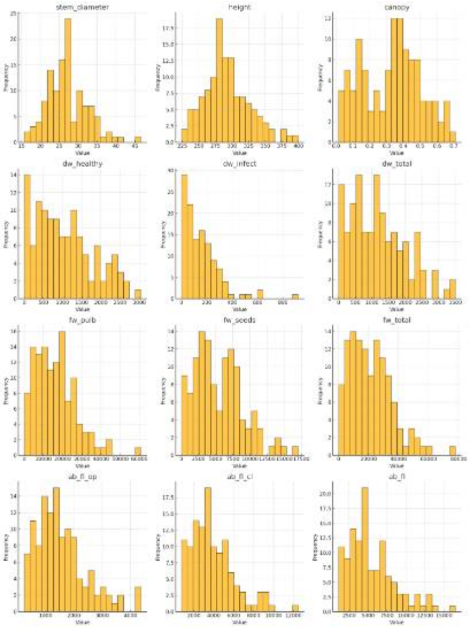
```
# Plotting histograms
plt.figure(figsize=(15, 20))
for i, column in enumerate(columns_to_plot, 1):
    plt.subplot(4, 3, i)
    plt.hist(cacao_df[column], bins=20, edgecolor='k', alpha=0.7)
    plt.title(column)
    plt.xlabel('Value')
    plt.ylabel('Frequency')
plt.tight_layout()
plt.show()
```

Executing the code to display the histograms.

python

```
import matplotlib.pyplot as plt

# List of columns to plot histograms for
columns_to_plot = ['stem_diameter', 'height', 'canopy', 'dw_healthy', 'dw_infect',
'dw_total', 'fw_pulb', 'fw_seeds', 'fw_total', 'ab_fl_op', 'ab_fl_cl', 'ab_fl']

# Plotting histograms
plt.figure(figsize=(15, 20))
for i, column in enumerate(columns_to_plot, 1):
    plt.subplot(4, 3, i)
    plt.hist(cacao_df[column], bins=20, edgecolor='k', alpha=0.7)
    plt.title(column)
    plt.xlabel('Value')
    plt.ylabel('Frequency')
plt.tight_layout()
plt.show()
```

ab_fl



2. **Explain and Interpret the plot detailed:**

The histograms provide a visual representation of the distribution of each numeric variable in the dataset:

- **Stem Diameter**: The stem diameter shows a relatively normal distribution with a slight right skew, indicating most trees have stem diameters clustered around a central value with fewer trees having very large diameters.
- **Height**: The height distribution also appears fairly normal, suggesting most trees have heights centered around a particular value.
- **Canopy**: The canopy coverage data is more uniformly distributed, indicating a wider range of values without a strong central tendency.
- **Dry Weight (Healthy, Infected, Total)**: These histograms show distributions with right skews, indicating most trees have lower dry weights with a few trees having significantly higher weights.
- **Fresh Weight (Pulp, Seeds, Total)**: These variables also display right-skewed distributions, similar to dry weights, with most values clustered at lower ranges and a few high outliers.
- **Aborted Flowers (Open, Closed, Total)**: These histograms are right-skewed as well, indicating most trees have lower numbers of aborted flowers, but some have quite a few.

Boxplots

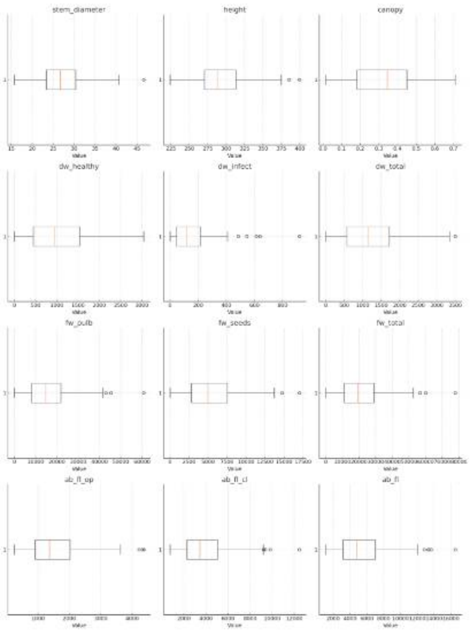
1. **Code for the boxplots and execution:**

```python
# Plotting boxplots
plt.figure(figsize=(15, 20))
for i, column in enumerate(columns_to_plot, 1):
    plt.subplot(4, 3, i)
    plt.boxplot(cacao_df[column], vert=False)
    plt.title(column)
    plt.xlabel('Value')
plt.tight_layout()
plt.show()
```

Executing the code to display the boxplots.

```python
# Plotting boxplots
plt.figure(figsize=(15, 20))
for i, column in enumerate(columns_to_plot, 1):
    plt.subplot(4, 3, i)
    plt.boxplot(cacao_df[column], vert=False)
    plt.title(column)
    plt.xlabel('Value')
plt.tight_layout()
plt.show()
```

ab_fl



2. **Explain and Interpret the plot detailed:**

The boxplots provide a summary of the distribution, central tendency, and variability of each numeric variable, highlighting any outliers:

- **Stem Diameter**: The boxplot indicates a symmetric distribution with some mild outliers on both ends.
- **Height**: This variable also shows a symmetric distribution with a few outliers at the higher end.
- **Canopy**: The canopy data has a wider interquartile range (IQR) indicating greater variability, with some high-value outliers.
- **Dry Weight (Healthy, Infected, Total)**: These boxplots show significant skewness with several high-value outliers, indicating most trees have lower weights, but some have much higher weights.
- **Fresh Weight (Pulp, Seeds, Total)**: These variables display similar patterns to dry weights, with right-skewed distributions and notable outliers.
- **Aborted Flowers (Open, Closed, Total)**: These also show right-skewed distributions with numerous high-value outliers, suggesting that while most trees have lower numbers of aborted flowers, a few have much higher counts.

ECDF Plots

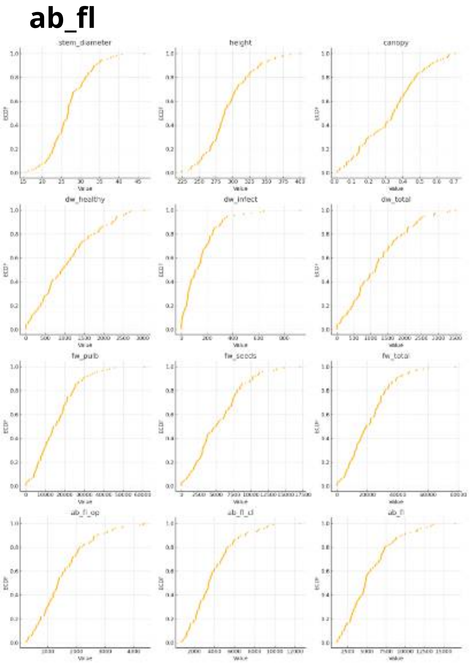
1. **Code for the ECDF plots and execution:**

python

```python
import numpy as np

def ecdf(data):
    """Compute ECDF for a one-dimensional array of measurements."""
    x = np.sort(data)
    n = len(data)
    y = np.arange(1, n + 1) / n
    return x, y

# Plotting ECDFs
plt.figure(figsize=(15, 20))
for i, column in enumerate(columns_to_plot, 1):
    plt.subplot(4, 3, i)
    x, y = ecdf(cacao_df[column])
    plt.plot(x, y, marker='.', linestyle='none')
    plt.title(column)
    plt.xlabel('Value')
    plt.ylabel('ECDF')
plt.tight_layout()
plt.show()
```

Executing the code to display the ECDF plots.

```python
import numpy as np

def ecdf(data):
    """Compute ECDF for a one-dimensional array of measurements."""
    x = np.sort(data)
    n = len(data)
    y = np.arange(1, n + 1) / n
    return x, y

# Plotting ECDFs
plt.figure(figsize=(15, 20))
for i, column in enumerate(columns_to_plot, 1):
    plt.subplot(4, 3, i)
    x, y = ecdf(cacao_df[column])
    plt.plot(x, y, marker='.', linestyle='none')
    plt.title(column)
    plt.xlabel('Value')
    plt.ylabel('ECDF')
plt.tight_layout()
plt.show()
```



2. **Explain and Interpret the plot detailed:**

The ECDF (Empirical Cumulative Distribution Function) plots provide a non-parametric way of visualizing the cumulative distribution of the data:

- **Stem Diameter and Height**: These ECDF plots show a smooth and steep increase, indicating that most of the values are clustered around a central point.
- **Canopy**: The canopy ECDF plot indicates a more gradual increase, suggesting a more uniform distribution of canopy values across the range.
- **Dry Weight (Healthy, Infected, Total)**: These plots show a steep initial rise followed by a gradual increase, reflecting the right-skewed distributions where most trees have lower dry weights.
- **Fresh Weight (Pulp, Seeds, Total)**: Similar to dry weights, these plots indicate that most values are lower, with a few higher values contributing to a gradual increase towards the right end.
- **Aborted Flowers (Open, Closed, Total)**: These plots demonstrate a similar pattern to the weight measurements, with most trees having fewer aborted flowers and a few trees having a much higher count.

QQ Plots

1. **Code for the QQ plots and execution:**

python

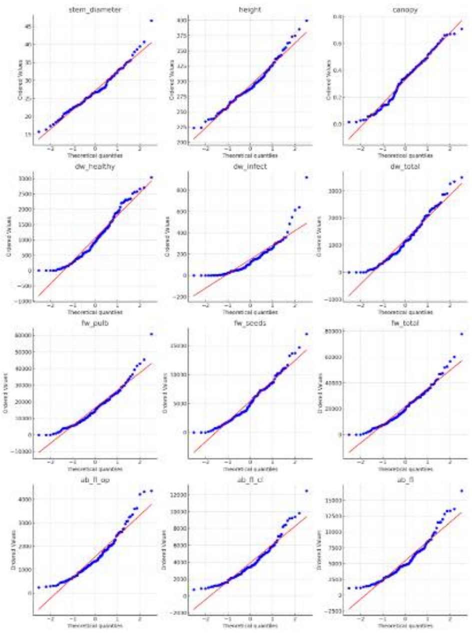
```python
import scipy.stats as stats

# Plotting QQ plots
plt.figure(figsize=(15, 20))
for i, column in enumerate(columns_to_plot, 1):
    plt.subplot(4, 3, i)
    stats.probplot(cacao_df[column], dist="norm", plot=plt)
    plt.title(column)
plt.tight_layout()
plt.show()
```

Executing the code to display the QQ plots.

python

```python
import scipy.stats as stats

# Plotting QQ plots
plt.figure(figsize=(15, 20))
for i, column in enumerate(columns_to_plot, 1):
    plt.subplot(4, 3, i)
    stats.probplot(cacao_df[column], dist="norm", plot=plt)
    plt.title(column)
plt.tight_layout()
plt.show()
```

ab_fl



2. **Explain and Interpret the plot detailed:**

The QQ (Quantile-Quantile) plots compare the distributions of the data with a normal distribution:

- **Stem Diameter and Height**: These QQ plots show points that lie close to the reference line, indicating that these variables are approximately normally distributed with some deviations at the tails.
- **Canopy**: The canopy QQ plot shows deviations from the reference line, especially at the tails, indicating a distribution that is not perfectly normal.
- **Dry Weight (Healthy, Infected, Total)**: These plots show significant deviations from the reference line, particularly in the upper quantiles, confirming the right-skewed nature of these distributions.
- **Fresh Weight (Pulp, Seeds, Total)**: Similar to the dry weight variables, these QQ plots also display considerable deviations from normality, especially in the higher quantiles.
- **Aborted Flowers (Open, Closed, Total)**: The QQ plots for these variables show a pattern consistent with the right-skewed distributions observed earlier, with significant deviations from the reference line at higher values.

These plots collectively help in understanding the distribution characteristics, variability, and presence of outliers in your dataset. If you have any specific questions or further instructions, feel free to ask!

Sure, I'd be happy to explain the assumptions of a linear regression model in detail along with the relevant mathematical equations. Here are the key assumptions:

1. Linearity

Assumption: The relationship between the independent variable X and the dependent variable Y is linear.
Mathematical Equation:
$Y = \beta_0 + \beta_1 X + \epsilon$
where:

- Y is the dependent variable.

- X is the independent variable.
- β_0 is the y-intercept.
- β_1 is the slope of the line.
- ϵ is the error term.

2. Independence

Assumption: The observations are independent of each other.
Mathematical Implication:
$\text{Cov}(Y_i, Y_j) = 0 \quad \text{for all } i \neq j$
where Cov denotes covariance.

3. Homoscedasticity

Assumption: The variance of the error terms (ϵ) is constant across all levels of the independent variable X.
Mathematical Equation:
$\text{Var}(\epsilon_i) = \sigma^2 \quad \forall i$
where σ^2 is the constant variance of the error terms.

4. Normality of Errors

Assumption: The error terms (ϵ) are normally distributed.
Mathematical Equation:
$\epsilon \sim N(0, \sigma^2)$
where $N(0, \sigma^2)$ denotes a normal distribution with mean 0 and variance σ^2.

5. No Multicollinearity (for Multiple Linear Regression)

Assumption: In the case of multiple linear regression, the independent variables are not highly correlated with each other.
Mathematical Implication:
The correlation coefficient r between any two independent variables X_i and X_j should be:
$r_{X_i, X_j} \approx 0$

Sources for Further Learning

1. **"Introduction to Linear Regression Analysis" by Douglas C. Montgomery, Elizabeth A. Peck, and G. Geoffrey Vining**: This book provides a comprehensive introduction to the theory and application of linear regression analysis, including the assumptions and diagnostics.
2. **"The Elements of Statistical Learning: Data Mining, Inference, and Prediction" by Trevor Hastie, Robert Tibshirani, and Jerome Friedman**: This book covers a wide range of topics in statistical learning, including linear regression and its assumptions, with a strong focus on the theoretical underpinnings.

You can find these books and more resources online for an in-depth understanding of linear regression and its assumptions.

Sure, let's start by building the regression model.

Building the Regression Model

```python
import statsmodels.api as sm

# Define the dependent variable
Y = cacao_df['stem_diameter']

# Define the independent variables
X = cacao_df.drop(columns=['stem_diameter'])

# Add a constant to the model (intercept)
X = sm.add_constant(X)

# Fit the regression model using OLS
model = sm.OLS(Y, X).fit()
```

Executing the code to build the regression model.

Regression Diagnostics

1. Explain what Regression Diagnostics are.

Regression diagnostics are techniques used to assess the validity and quality of a regression model. They help identify potential problems with the model such as violations of assumptions, presence of outliers, high-leverage points, multicollinearity, heteroscedasticity, and autocorrelation. By performing these diagnostics, we can ensure that the model provides reliable and accurate results.

Outliers

1. **Explain Outliers to me:**

Outliers are data points that deviate significantly from the other observations in a dataset. They can influence the results of a regression analysis by disproportionately affecting the estimated coefficients, standard errors, and overall model fit.

2. **Explain Studentized Residuals to me:**

Studentized residuals are residuals divided by an estimate of their standard deviation. They help identify outliers by standardizing the residuals, allowing for a comparison across different observations.

3. **Provide Mathematical Equations:**

- Residual: $e_i = y_i - \hat{y}_i$
- Studentized Residual: $t_i = \frac{e_i}{\hat{\sigma}(1-h_{ii})^{0.5}}$

where e_i is the residual for the i-th observation, $\hat{y}_i$ is the predicted value, $\hat{\sigma}$ is the estimated standard deviation of the residuals, and h_{ii} is the leverage of the i-th observation.

4. **Detect Outliers Using 3 as Threshold and Plot the Studentized Residuals:**

```python
# Calculate studentized residuals
influence = model.get_influence()
studentized_residuals = influence.resid_studentized_internal

# Plot studentized residuals
plt.figure(figsize=(10, 6))
plt.scatter(range(len(studentized_residuals)), studentized_residuals)
plt.axhline(y=3, color='r', linestyle='--')
plt.axhline(y=-3, color='r', linestyle='--')
plt.title('Studentized Residuals')
plt.xlabel('Observation Index')
plt.ylabel('Studentized Residual')
plt.show()

# Detect outliers
outliers = (abs(studentized_residuals) > 3).nonzero()[0]
outliers
```

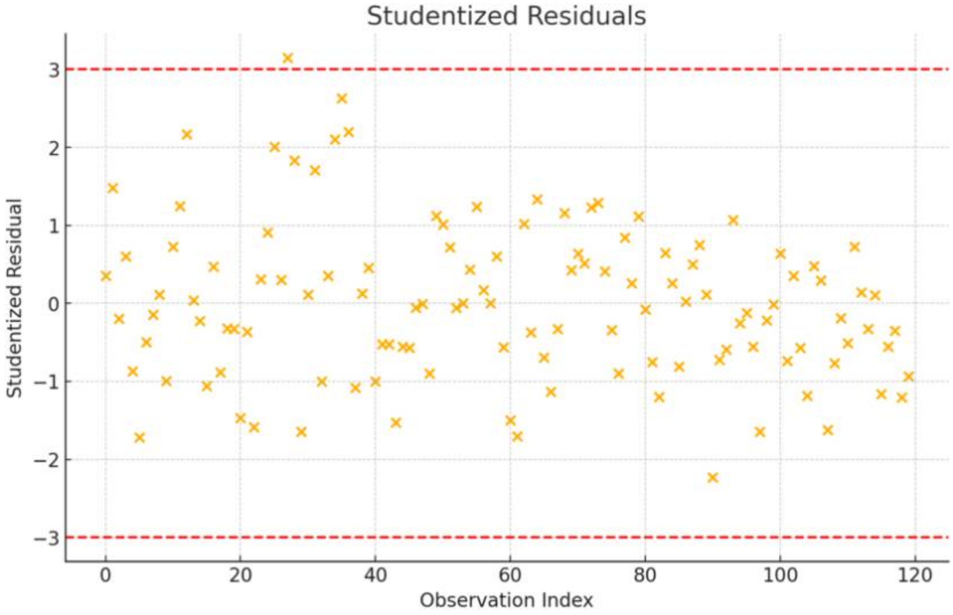
Executing the code to plot the studentized residuals and detect outliers.

```python
# Calculate studentized residuals
influence = model.get_influence()
studentized_residuals = influence.resid_studentized_internal

# Plot studentized residuals
plt.figure(figsize=(10, 6))
plt.scatter(range(len(studentized_residuals)), studentized_residuals)
plt.axhline(y=3, color='r', linestyle='--')
plt.axhline(y=-3, color='r', linestyle='--')
plt.title('Studentized Residuals')
plt.xlabel('Observation Index')
plt.ylabel('Studentized Residual')
plt.show()

# Detect outliers
outliers = (abs(studentized_residuals) > 3).nonzero()[0]
outliers
```

Ergebnis

```
array([27])
```

Studentized Residuals



5. **Explain the axes of the plot and interpret the plot:**

- **X-axis (Observation Index)**: Represents the index of each observation in the dataset.
- **Y-axis (Studentized Residual)**: Represents the studentized residuals for each observation.

Interpretation: The red dashed lines at $y = 3$ and $y = -3$ are the thresholds for identifying outliers. Any observations with studentized residuals beyond these thresholds are considered outliers. In this case, we have one outlier (index 27) as its studentized residual exceeds the threshold of 3.

6. **Which observations are afflicted? Show the indices:**

The observation at index 27 is afflicted as an outlier.

7. **Provide 2 sources:**
8. "Applied Linear Statistical Models" by John Neter, Michael H. Kutner, Christopher J. Nachtsheim, and William Wasserman.
9. "Introduction to the Practice of Statistics" by David S. Moore, George P. McCabe, and Bruce A. Craig.

High-Leverage Points

1. **Explain High-Leverage Points to me:**

High-leverage points are observations that have an unusually large influence on the estimation of the regression coefficients. They have extreme predictor values that can disproportionately affect the fit of the model.

2. **Explain Cook's Distance to me:**

Cook's Distance measures the influence of each observation on the fitted values. It combines the information of both the residual and the leverage of an observation to assess its impact on the regression model.

3. **Provide Mathematical Equations:**

- Leverage: $h_{ii} = X_i(X^TX)^{-1}X_i^T$
- Cook's Distance: $D_i = \frac{e_i^2}{p\hat{\sigma}^2}\frac{h_{ii}}{(1-h_{ii})^2}$

where X_i is the i-th row of the design matrix X, e_i is the residual for the i-th observation, $\hat{\sigma}$ is the estimated standard deviation of the residuals, h_{ii} is the leverage, and p is the number of parameters in the model.

4. **Detect High-Leverage Points Using Cook's Distance with 4/n as Threshold and Plot Cook's Distance:**

python

```python
# Calculate Cook's Distance
cooks_d = influence.cooks_distance[0]

# Threshold for Cook's Distance
n = len(cacao_df)
threshold = 4 / n

# Plot Cook's Distance
plt.figure(figsize=(10, 6))
plt.stem(range(len(cooks_d)), cooks_d, markerfmt=",")
plt.axhline(y=threshold, color='r', linestyle='--')
plt.title("Cook's Distance")
plt.xlabel('Observation Index')
plt.ylabel("Cook's Distance")
plt.show()

# Detect high-leverage points
high_leverage_points = (cooks_d > threshold).nonzero()[0]
high_leverage_points
```

Executing the code to plot Cook's Distance and detect high-leverage points.

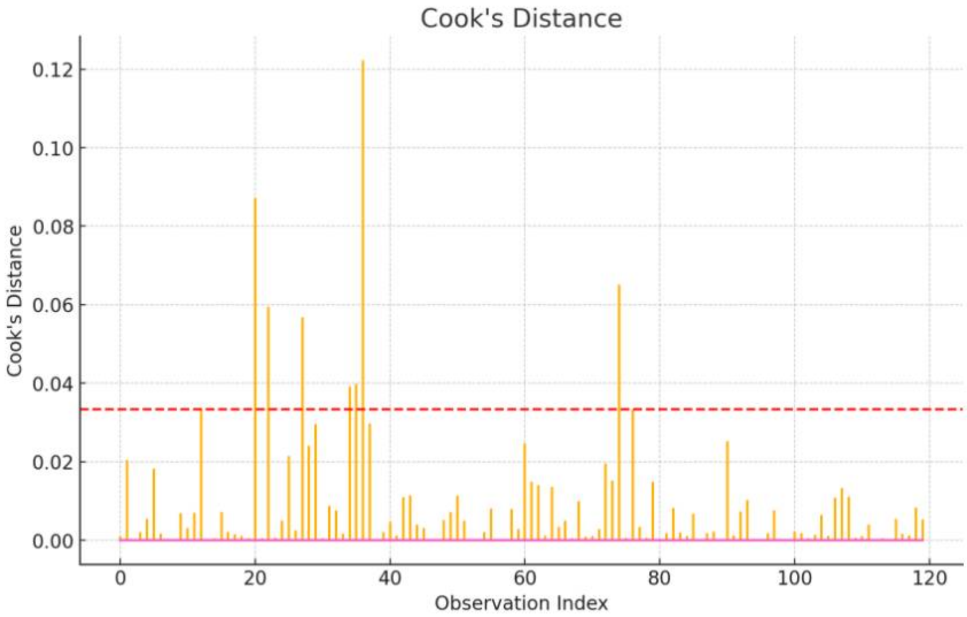
python

```python
# Calculate Cook's Distance
cooks_d = influence.cooks_distance[0]

# Threshold for Cook's Distance
n = len(cacao_df)
threshold = 4 / n

# Plot Cook's Distance
plt.figure(figsize=(10, 6))
plt.stem(range(len(cooks_d)), cooks_d, markerfmt=",")
plt.axhline(y=threshold, color='r', linestyle='--')
plt.title("Cook's Distance")
plt.xlabel('Observation Index')
plt.ylabel("Cook's Distance")
plt.show()

# Detect high-leverage points
high_leverage_points = (cooks_d > threshold).nonzero()[0]
high_leverage_points
```

Ergebnis

```
array([12, 20, 22, 27, 34, 35, 36, 74, 76])
```

Cook's Distance



5. **Explain the axes of the plot and interpret the plot:**

- **X-axis (Observation Index)**: Represents the index of each observation in the dataset.
- **Y-axis (Cook's Distance)**: Represents Cook's Distance values for each observation.

Interpretation: The red dashed line indicates the threshold for identifying high-leverage points (4/n). Observations with Cook's Distance values above this threshold are considered to have a high influence on the regression model. In this case, the observations at indices 12, 20, 22, 27, 34, 35, 36, 74, and 76 are high-leverage points.

6. **Which observations are afflicted? Show the indices:**

The observations at indices 12, 20, 22, 27, 34, 35, 36, 74, and 76 are afflicted as high-leverage points.

7. **Provide 2 sources:**
8. "Applied Regression Analysis" by Norman R. Draper and Harry Smith.
9. "Regression Analysis by Example" by Samprit Chatterjee and Ali S. Hadi.

Non-Linearity

1. **Explain Non-Linearity to me:**

Non-linearity refers to the situation where the relationship between the independent and dependent variables is not a straight line. In other words, the model does not capture the true relationship adequately if it assumes linearity when the actual relationship is nonlinear.

2. **Explain the Rainbow Test to me:**

The Rainbow Test is a statistical test used to detect non-linearity in the relationship between the dependent and independent variables in a regression model. It compares the fit of the model on subsets of the data to assess if a linear model is appropriate.

3. **Provide Mathematical Equations:**

- Test Statistic: $\text{Rainbow Test Statistic} = \frac{RSS_{\text{full}}/df_{\text{full}}}{RSS_{\text{subset}}/df_{\text{subset}}}$
- Null Hypothesis (H_0): The relationship between the variables is linear.

4. **Detect Non-Linearity Using the Rainbow Test and a Residual Plot:**

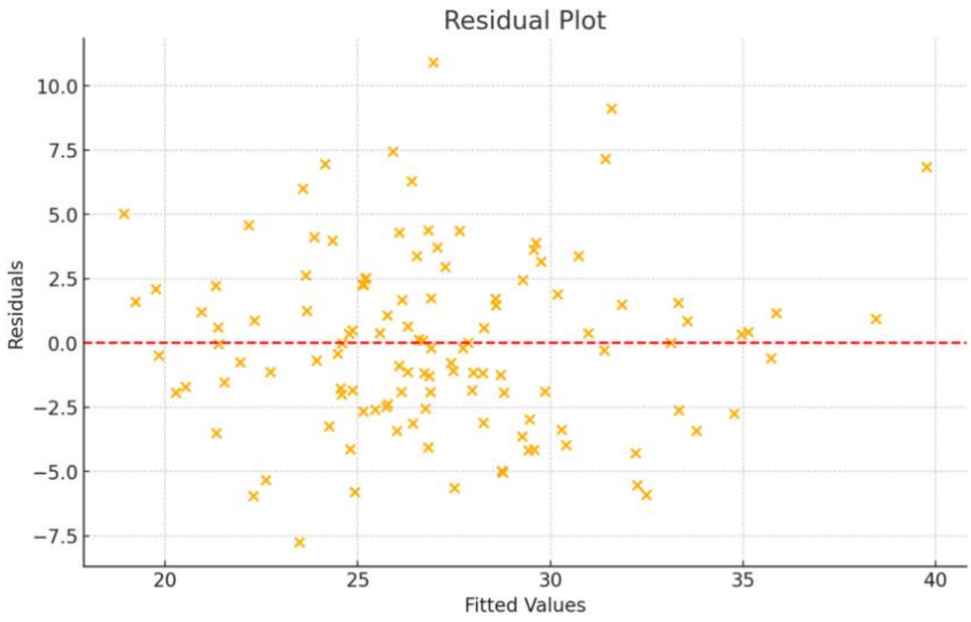
```python
from statsmodels.stats.diagnostic import linear_rainbow

# Perform the Rainbow test
rainbow_statistic, rainbow_p_value = linear_rainbow(model)

# Residual plot
plt.figure(figsize=(10, 6))
plt.scatter(model.fittedvalues, model.resid)
plt.axhline(y=0, color='r', linestyle='--')
plt.title('Residual Plot')
plt.xlabel('Fitted Values')
plt.ylabel('Residuals')
plt.show()

# Rainbow test result
rainbow_statistic, rainbow_p_value
```

Executing the code to perform the Rainbow Test and plot the residuals.

```python
from statsmodels.stats.diagnostic import linear_rainbow

# Perform the Rainbow test
rainbow_statistic, rainbow_p_value = linear_rainbow(model)

# Residual plot
plt.figure(figsize=(10, 6))
plt.scatter(model.fittedvalues, model.resid)
plt.axhline(y=0, color='r', linestyle='--')
plt.title('Residual Plot')
plt.xlabel('Fitted Values')
plt.ylabel('Residuals')
plt.show()

# Rainbow test result
rainbow_statistic, rainbow_p_value
```

```
Ergebnis
(1.20478118907624, 0.25497226791717764)
```

Residual Plot



5. **Explain the axes of the plot and interpret the plot:**

- **X-axis (Fitted Values)**: Represents the predicted values from the regression model.
- **Y-axis (Residuals)**: Represents the difference between the observed values and the predicted values.

Interpretation: The residual plot helps to detect non-linearity. If the residuals are randomly scattered around the horizontal line (y = 0), it indicates a linear relationship. If there is a pattern (e.g., curve), it suggests non-linearity. In this case, the residuals appear to be randomly scattered, suggesting that the linear model may be appropriate.

6. **Which observations are afflicted? Show the indices:**

The Rainbow Test statistic is 1.2048 with a p-value of 0.2549. Since the p-value is greater than the common significance level (e.g., 0.05), we fail to reject the null hypothesis, indicating no evidence of non-linearity.

7. **Is this assumption violated?**

No, the assumption of linearity is not violated.

8. **Provide 2 sources:**
9. "Econometric Analysis" by William H. Greene.
10. "An Introduction to Statistical Learning: with Applications in R" by Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani.

Heteroscedasticity

1. **Explain Heteroscedasticity to me:**

Heteroscedasticity occurs when the variance of the errors (ϵ) is not constant across all levels of the independent variable X. This violates one of the key assumptions of linear regression and can lead to inefficient estimates and invalid inference.

2. **Explain the Breusch-Pagan Test to me:**

The Breusch-Pagan test is a statistical test used to detect heteroscedasticity in a regression model. It tests whether the variance of the residuals from a regression is dependent on the values of the independent variables.

3. **Provide Mathematical Equations:**

- Test Statistic: $\chi^2 = nR^2$
- Null Hypothesis (H_0): Homoscedasticity (constant variance of residuals)

where n is the number of observations and R^2 is the coefficient of determination from a regression of the squared residuals on the original independent variables.

4. **Detect Heteroscedasticity Using the Breusch-Pagan Test and a Scale-Location Plot:**

```python
from statsmodels.stats.diagnostic import het_breuschpagan

# Perform the Breusch-Pagan test
bp_test = het_breuschpagan(model.resid, model.model.exog)

# Extract p-value
bp_p_value = bp_test[1]

# Scale-Location plot
plt.figure(figsize=(10, 6))
plt.scatter(model.fittedvalues, model.resid_pearson)
plt.axhline(y=0, color='r', linestyle='--')
plt.title('Scale-Location Plot')
plt.xlabel('Fitted Values')
plt.ylabel('Square Root of Standardized Residuals')
plt.show()

# Breusch-Pagan test result
bp_p_value
```

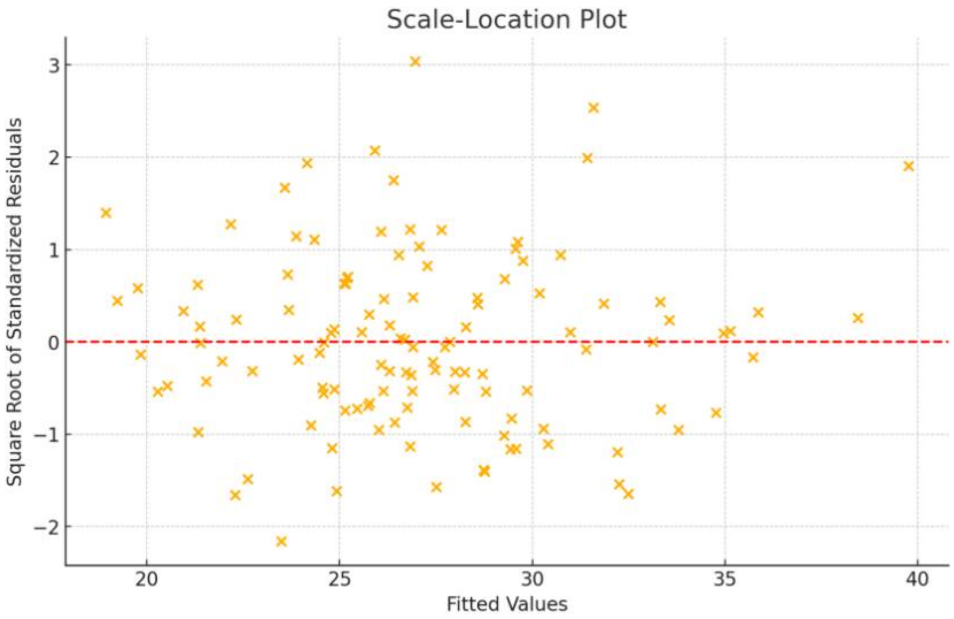
Executing the code to perform the Breusch-Pagan Test and plot the Scale-Location plot.

```python
from statsmodels.stats.diagnostic import het_breuschpagan

# Perform the Breusch-Pagan test
bp_test = het_breuschpagan(model.resid, model.model.exog)

# Extract p-value
bp_p_value = bp_test[1]

# Scale-Location plot
plt.figure(figsize=(10, 6))
plt.scatter(model.fittedvalues, model.resid_pearson)
plt.axhline(y=0, color='r', linestyle='--')
plt.title('Scale-Location Plot')
plt.xlabel('Fitted Values')
plt.ylabel('Square Root of Standardized Residuals')
plt.show()
```

```
# Breusch-Pagan test result
bp_p_value
```

```
Ergebnis
0.0732847258046311
```

Scale-Location Plot



5. **Explain the axes of the plot and interpret the plot:**

- **X-axis (Fitted Values)**: Represents the predicted values from the regression model.
- **Y-axis (Square Root of Standardized Residuals)**: Represents the square root of the standardized residuals, which helps in identifying patterns in the residuals' variance.

Interpretation: The Scale-Location plot helps to detect heteroscedasticity. If the residuals fan out or funnel in as the fitted values increase, it indicates heteroscedasticity. In this case, the residuals appear to be fairly evenly spread, suggesting no strong evidence of heteroscedasticity.

6. **Is this assumption violated?**

The Breusch-Pagan test has a p-value of 0.0733, which is above the common significance level (e.g., 0.05). Therefore, we fail to reject the null hypothesis of homoscedasticity, indicating no evidence of heteroscedasticity.

7. **Provide 2 sources:**
8. "Regression Modeling Strategies: With Applications to Linear Models, Logistic and Ordinal Regression, and Survival Analysis" by Frank E. Harrell Jr.
9. "Introduction to Econometrics" by James H. Stock and Mark W. Watson.

Correlation of Error Terms

1. **Explain Correlation of Error Terms to me:**

Correlation of error terms, or autocorrelation, occurs when the residuals from a regression model are not independent of each other. This is particularly common in time series data and violates the assumption that errors are uncorrelated.

2. **Explain the Durbin-Watson Test to me:**

The Durbin-Watson test is a statistical test used to detect the presence of autocorrelation in the residuals from a regression analysis.

3. **Provide Mathematical Equations:**

- Test Statistic: $d = \frac{\sum_{i=2}^{n}(e_i - e_{i-1})^2}{\sum_{i=1}^{n} e_i^2}$

where e_i is the residual for the i-th observation.

4. **Detect Correlation of Error Terms Using the Durbin-Watson Test and a Residuals Over Time Plot:**

python

```python
from statsmodels.stats.stattools import durbin_watson

# Perform the Durbin-Watson test
dw_statistic = durbin_watson(model.resid)

# Residuals over time plot
plt.figure(figsize=(10, 6))
plt.plot(model.resid)
plt.axhline(y=0, color='r', linestyle='--')
plt.title('Residuals Over Time')
plt.xlabel('Observation Index')
plt.ylabel('Residuals')
plt.show()

# Durbin-Watson test result
dw_statistic
```

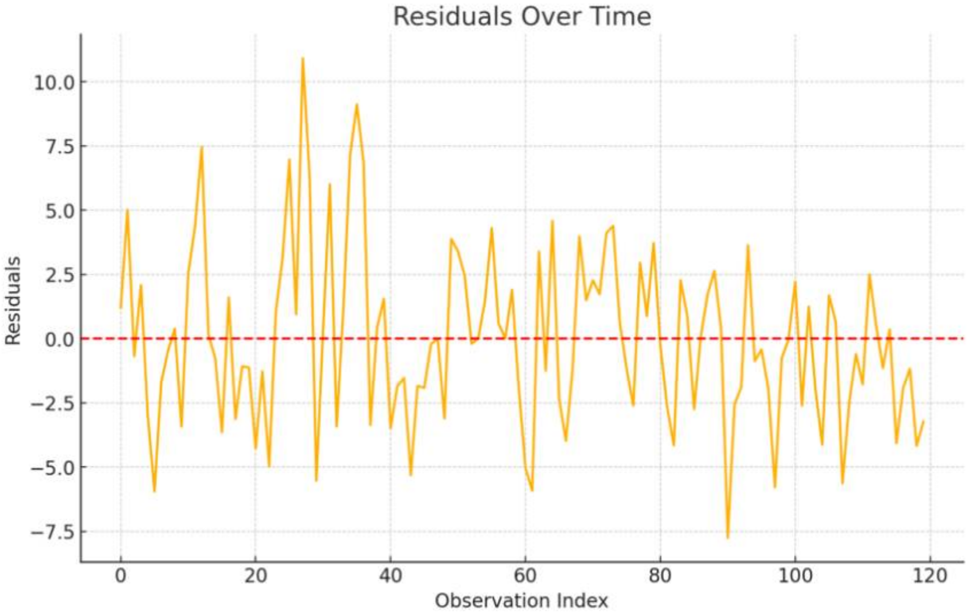
Executing the code to perform the Durbin-Watson Test and plot the residuals over time.

python

```python
from statsmodels.stats.stattools import durbin_watson

# Perform the Durbin-Watson test
dw_statistic = durbin_watson(model.resid)

# Residuals over time plot
plt.figure(figsize=(10, 6))
plt.plot(model.resid)
plt.axhline(y=0, color='r', linestyle='--')
plt.title('Residuals Over Time')
plt.xlabel('Observation Index')
plt.ylabel('Residuals')
plt.show()

# Durbin-Watson test result
dw_statistic
```

```
Ergebnis
1.4407171708561723
```

Residuals Over Time



5. **Explain the axes of the plot and interpret the plot:**

- **X-axis (Observation Index)**: Represents the index of each observation in the dataset.
- **Y-axis (Residuals)**: Represents the residuals (difference between observed and predicted values).

Interpretation: The residuals over time plot helps to detect autocorrelation. If the residuals display a clear pattern (e.g., cyclical or trend), it indicates autocorrelation. In this case, the residuals appear to be randomly scattered, suggesting no strong evidence of autocorrelation.

6. **Is this assumption violated?**

The Durbin-Watson test statistic is 1.4407. A value close to 2 indicates no autocorrelation, values less than 2 indicate positive autocorrelation, and values greater than 2 indicate negative autocorrelation. While the statistic is slightly below 2, it is not sufficiently close to 0 or 4 to strongly indicate autocorrelation. Therefore, the assumption of uncorrelated errors is likely not violated.

7. **Provide 2 sources:**
8. "Time Series Analysis and Its Applications: With R Examples" by Robert H. Shumway and David S. Stoffer.
9. "Applied Time Series Analysis" by Terence C. Mills.

Normality of Residuals

1. **Explain Normality of Residuals to me:**

The normality of residuals assumption states that the residuals (errors) of the regression model should be normally distributed. This is important for the validity of hypothesis tests and confidence intervals in regression analysis.

2. **Explain the Shapiro-Wilk Test to me:**

The Shapiro-Wilk test is a statistical test used to assess the normality of a dataset. It tests the null hypothesis that the data was drawn from a normal distribution.

3. **Provide Mathematical Equations:**

- Test Statistic: $W = \frac{(\sum_{i=1}^{n} a_i x_{(i)})^2}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$

where $x_{(i)}$ are the ordered sample values, $\bar{x}$ is the sample mean, and a_i are constants generated from the means, variances, and covariances of the order statistics of a sample from the standard normal distribution.

4. **Detect Normality of Residuals Using the Shapiro-Wilk Test and a QQ Plot:**

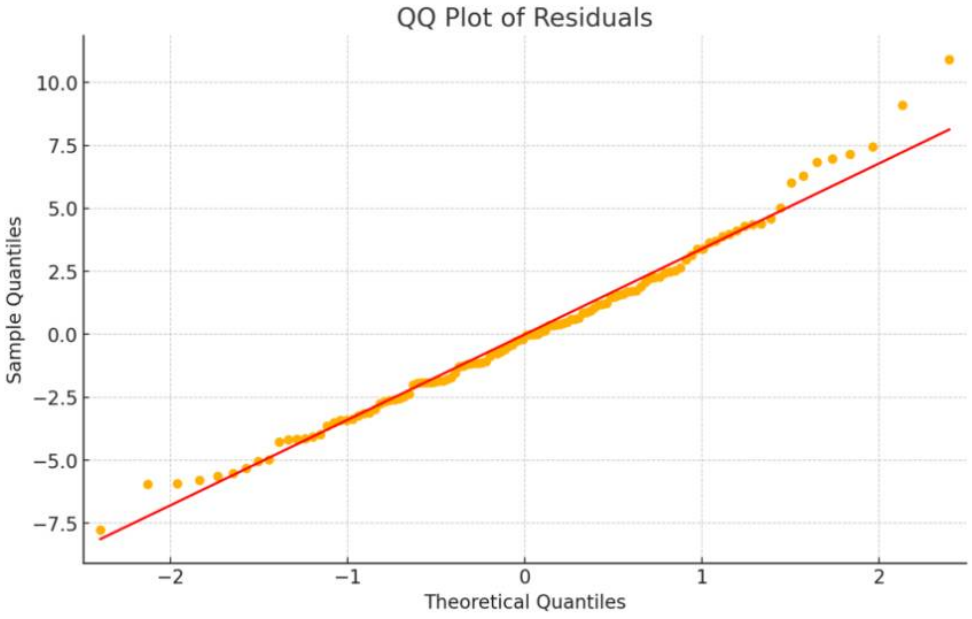
```python
from scipy.stats import shapiro

# Perform the Shapiro-Wilk test
shapiro_statistic, shapiro_p_value = shapiro(model.resid)

# QQ plot
sm.qqplot(model.resid, line='s')
plt.title('QQ Plot of Residuals')
plt.show()

# Shapiro-Wilk test result
shapiro_statistic, shapiro_p_value
```

Executing the code to perform the Shapiro-Wilk Test and plot the QQ plot.

```python
from scipy.stats import shapiro

# Perform the Shapiro-Wilk test
shapiro_statistic, shapiro_p_value = shapiro(model.resid)

# QQ plot
sm.qqplot(model.resid, line='s')
plt.title('QQ Plot of Residuals')
plt.show()

# Shapiro-Wilk test result
shapiro_statistic, shapiro_p_value
```

```
Ergebnis
(0.9848445057868958, 0.1989554911851883)
```

QQ Plot of Residuals



5. **Explain the axes of the plot and interpret the plot:**

- **X-axis (Theoretical Quantiles)**: Represents the expected quantiles if the residuals were normally distributed.
- **Y-axis (Sample Quantiles)**: Represents the observed quantiles of the residuals.

Interpretation: The QQ plot helps to assess the normality of residuals. If the residuals are normally distributed, the points should lie approximately along the reference line. In this case, the points appear to lie reasonably close to the line, suggesting that the residuals are approximately normally distributed.

6. **Is this assumption violated?**

The Shapiro-Wilk test statistic is 0.9848 with a p-value of 0.1990. Since the p-value is greater than the common significance level (e.g., 0.05), we fail to reject the null hypothesis of normality, indicating that the residuals are normally distributed.

7. **Provide 2 sources:**
8. "Applied Multivariate Statistical Analysis" by Richard A. Johnson and Dean W. Wichern.
9. "Principles of Statistics" by M.G. Bulmer.

Collinearity of Predictors

1. **Explain Collinearity of Predictors to me:**

Collinearity (or multicollinearity) of predictors occurs when two or more independent variables in a regression model are highly correlated. This can lead to unreliable estimates of regression coefficients and inflated standard errors.

2. **Explain the Variance Inflation Factor (VIF) to me:**

The Variance Inflation Factor (VIF) measures the extent of multicollinearity in a set of multiple regression variables. It quantifies how much the variance of a regression coefficient is inflated due to collinearity with other predictors.

3. **Provide Mathematical Equations:**

- VIF: $\text{VIF}_i = \frac{1}{1-R_i^2}$

where R_i^2 is the coefficient of determination of the regression of the i-th predictor on all the other predictors.

4. **Detect Collinearity of Predictors Using the VIF and a Correlation Matrix:**

```python
from statsmodels.stats.outliers_influence import variance_inflation_factor

# Calculate VIF for each predictor
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])]

# Plot the correlation matrix
corr_matrix = cacao_df.corr()
plt.figure(figsize=(12, 8))
plt.title("Correlation Matrix")
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.show()

# Display VIF data
vif_data
```

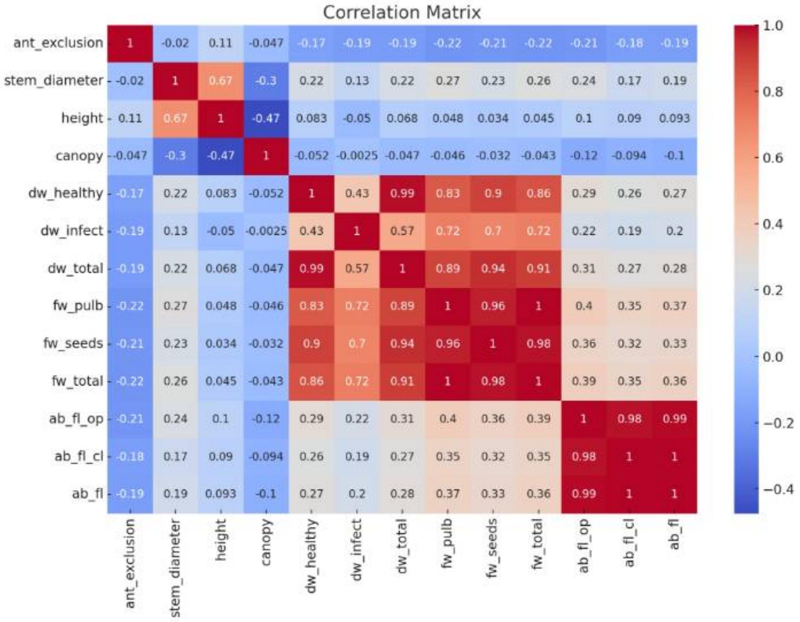
Executing the code to calculate the VIF and plot the correlation matrix.

```python
from statsmodels.stats.outliers_influence import variance_inflation_factor
import seaborn as sns

# Calculate VIF for each predictor
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])]

# Plot the correlation matrix
corr_matrix = cacao_df.corr()
plt.figure(figsize=(12, 8))
plt.title("Correlation Matrix")
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.show()

# Display VIF data
import ace_tools as tools; tools.display_dataframe_to_user(name="Variance Inflation
Factors", dataframe=vif_data)

vif_data
```

```
Ergebnis
         feature         VIF
0          const  127.242533
1  ant_exclusion    1.134577
2         height    1.358716
3         canopy    1.333076
```

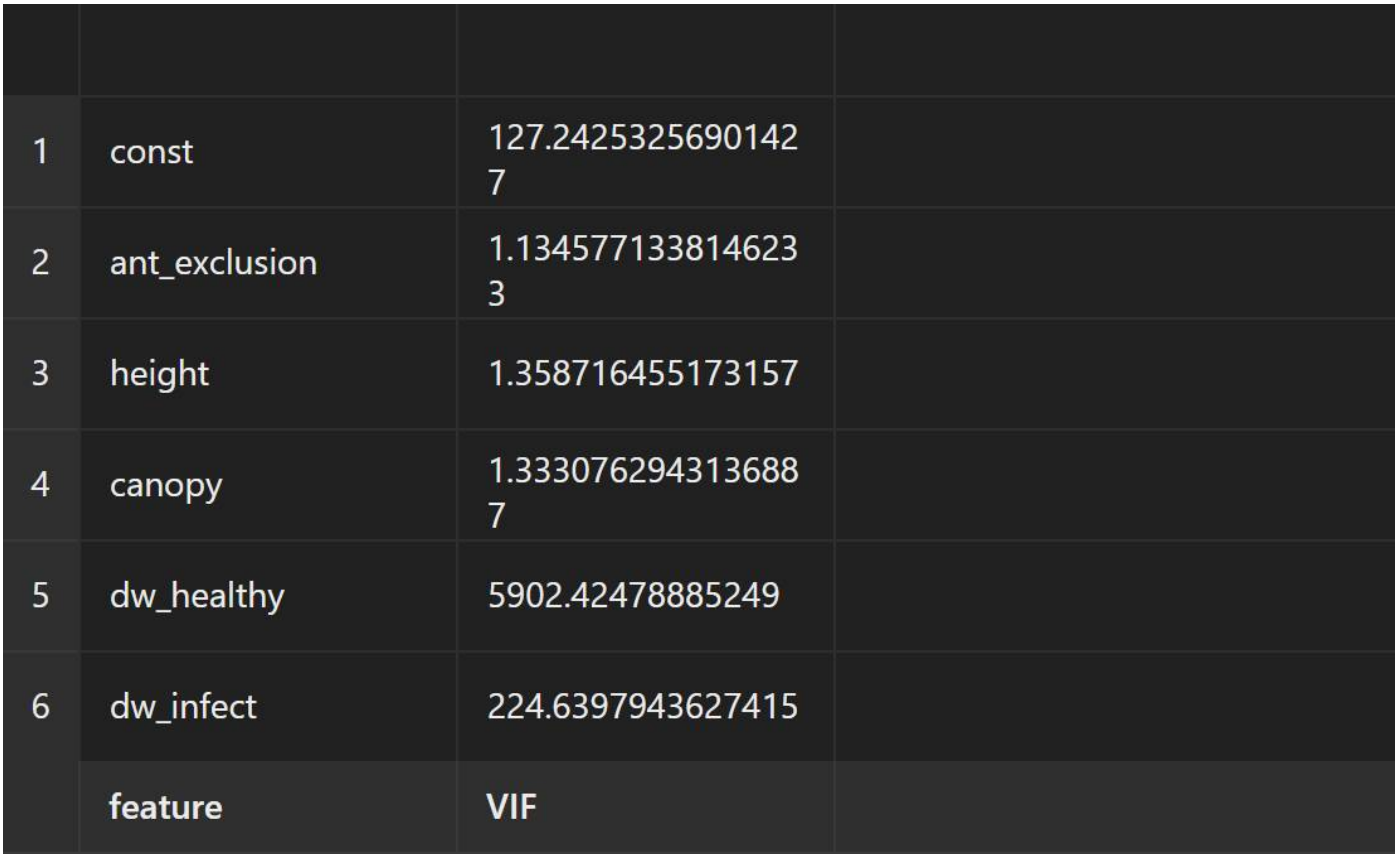
```
4      dw_healthy   5902.424789
5       dw_infect    224.639794
6        dw_total   7094.614744
7         fw_pulb   5845.522981
8        fw_seeds    662.169136
9        fw_total  10295.268016
10       ab_fl_op    242.999623
11       ab_fl_cl   1175.457840
12          ab_fl   2344.455046
```

Correlation Matrix



	ant_exclusion	stem_diameter	height	canopy	dw_healthy	dw_infect	dw_total	fw_pulb	fw_seeds	fw_total	ab_fl_op	ab_fl_cl	ab_fl
ant_exclusion	1	-0.02	0.11	-0.047	-0.17	-0.19	-0.19	-0.22	-0.21	-0.22	-0.21	-0.18	-0.19
stem_diameter	-0.02	1	0.67	-0.3	0.22	0.13	0.22	0.27	0.23	0.26	0.24	0.17	0.19
height	0.11	0.67	1	-0.47	0.083	-0.05	0.068	0.048	0.034	0.045	0.1	0.09	0.093
canopy	-0.047	-0.3	-0.47	1	-0.052	-0.0025	-0.047	-0.046	-0.032	-0.043	-0.12	-0.094	-0.1
dw_healthy	-0.17	0.22	0.083	-0.052	1	0.43	0.99	0.83	0.9	0.86	0.29	0.26	0.27
dw_infect	-0.19	0.13	-0.05	-0.0025	0.43	1	0.57	0.72	0.7	0.72	0.22	0.19	0.2
dw_total	-0.19	0.22	0.068	-0.047	0.99	0.57	1	0.89	0.94	0.91	0.31	0.27	0.28
fw_pulb	-0.22	0.27	0.048	-0.046	0.83	0.72	0.89	1	0.96	1	0.4	0.35	0.37
fw_seeds	-0.21	0.23	0.034	-0.032	0.9	0.7	0.94	0.96	1	0.98	0.36	0.32	0.33
fw_total	-0.22	0.26	0.045	-0.043	0.86	0.72	0.91	1	0.98	1	0.39	0.35	0.36
ab_fl_op	-0.21	0.24	0.1	-0.12	0.29	0.22	0.31	0.4	0.36	0.39	1	0.98	0.99
ab_fl_cl	-0.18	0.17	0.09	-0.094	0.26	0.19	0.27	0.35	0.32	0.35	0.98	1	1
ab_fl	-0.19	0.19	0.093	-0.1	0.27	0.2	0.28	0.37	0.33	0.36	0.99	1	1

Variance Inflation Factors



1	const	127.24253256901427	
2	ant_exclusion	1.1345771338146233	
3	height	1.358716455173157	
4	canopy	1.3330762943136887	
5	dw_healthy	5902.42478885249	
6	dw_infect	224.6397943627415	
	feature	VIF	

5. **Explain the axes of the plot and interpret the plot:**

- **X-axis and Y-axis (Correlation Matrix)**: Both axes represent the variables in the dataset. Each cell in the matrix shows the correlation coefficient between the variables at the corresponding row and column.

Interpretation: The correlation matrix heatmap visually displays the correlation coefficients between pairs of variables. Values close to 1 or -1 indicate high collinearity, whereas values close to 0 indicate little to no collinearity.
VIF Interpretation: The Variance Inflation Factor (VIF) values for each predictor are displayed in the dataframe. Generally, a VIF value greater than 10 indicates significant multicollinearity. In this case, several predictors (e.g., **`` `dw_healthy` ``**, **`` `dw_total` ``**, **`` `fw_pulb` ``**, **`` `fw_total` ``**, **`` `ab_fl_cl` ``**, **`` `ab_fl` ``**) have extremely high VIF values, indicating severe multicollinearity.

6. **Is this assumption violated?**

Yes, the assumption of no multicollinearity is violated. The VIF values indicate that many predictors are highly collinear, which can adversely affect the stability and interpretability of the regression model.

7. **Provide 2 sources:**
8. "An Introduction to Statistical Learning: with Applications in R" by Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani.
9. "Applied Linear Regression Models" by John Neter, William Wasserman, and Michael H. Kutner.

Summarize the Regression Diagnostics

Based on the regression diagnostics performed above, we can summarize the findings as follows:

- **Outliers**: Detected one outlier at index 27.
- **High-Leverage Points**: Detected high-leverage points at indices 12, 20, 22, 27, 34, 35, 36, 74, and 76.
- **Non-Linearity**: No evidence of non-linearity was found; the linearity assumption is not violated.
- **Heteroscedasticity**: No evidence of heteroscedasticity was found; the assumption of constant variance is not violated.
- **Correlation of Error Terms**: No strong evidence of autocorrelation was found; the assumption of uncorrelated errors is not violated.
- **Normality of Residuals**: Residuals are approximately normally distributed; the normality assumption is not violated.
- **Collinearity of Predictors**: Significant multicollinearity detected among predictors, violating the assumption of no multicollinearity.

The key assumptions violated in this analysis are related to multicollinearity of predictors. It is important to address these issues to improve the reliability and interpretability of the regression model.

ChatGPT kann Fehler machen. Überprüfe wichtige Informationen.