Certainly! Let's begin by explaining the process of statistical analysis you would perform on the dataset titled **Rice Cammeo and Osmancik** from UCI's Machine Learning Repository. I'll outline the steps you would follow, assuming you've already loaded the dataset into a suitable environment (like Python or R) for analysis.

The dataset contains features related to rice samples, specifically focusing on **Cammeo** and **Osmancik** varieties. We'll focus on two major tests: **Normal Distribution** analysis and **ANOVA Test**.

**1. Normal Distribution Test**

A Normal Distribution test helps us determine whether the data points in the dataset follow a bell-shaped normal distribution, which is important for making certain assumptions in further statistical tests.

**Steps:**

1. **Visual Inspection:**
   * **Histogram**: Plot histograms for each numerical feature in the dataset. If the data follows a normal distribution, the histogram will resemble a bell curve.
   * **Q-Q Plot**: This plot compares the quantiles of the data with the quantiles of a standard normal distribution. If the points on the Q-Q plot lie along a straight line, it suggests that the data is approximately normally distributed.
2. **Shapiro-Wilk Test**:
   * Perform the **Shapiro-Wilk test** for normality, which tests the null hypothesis that the data is normally distributed. The p-value of this test will indicate whether the data follows a normal distribution or not.
   * **Null Hypothesis (H₀):** The data is normally distributed.
   * **Alternative Hypothesis (H₁):** The data is not normally distributed.

Example in Python:

from scipy.stats import shapiro

# Apply Shapiro-Wilk test on a feature (e.g., 'feature\_1')

stat, p\_value = shapiro(dataset['feature\_1'])

if p\_value > 0.05:

print("Data follows normal distribution (fail to reject H₀)")

else:

print("Data does not follow normal distribution (reject H₀)")

1. **Kolmogorov-Smirnov Test**:
   * Another test you can apply is the **Kolmogorov-Smirnov test**, which compares the empirical distribution function of the sample to the cumulative distribution function of the normal distribution.

**2. ANOVA Test**

The **ANOVA (Analysis of Variance)** test is used to determine if there are any statistically significant differences between the means of three or more independent groups. In this case, you can perform an ANOVA to compare the means of different features (such as **Cammeo** and **Osmancik** rice varieties).

**Steps:**

1. **Hypotheses:**
   * **Null Hypothesis (H₀):** There is no significant difference in the means between the groups (e.g., no difference in the rice varieties' features).
   * **Alternative Hypothesis (H₁):** At least one group mean is significantly different from the others.
2. **Performing ANOVA:**
   * First, check if the assumptions for ANOVA are met:
     + Normality: Each group (e.g., each rice variety) should follow a normal distribution (check with Shapiro-Wilk or visual inspection).
     + Homogeneity of variance: The variance within each group should be approximately equal.
   * If the assumptions are met, you can proceed with ANOVA. If not, you might consider using non-parametric tests such as the Kruskal-Wallis test.
3. **One-Way ANOVA**:
   * Perform a **one-way ANOVA** to compare the means of the features between the two rice varieties (Cammeo and Osmancik).
   * Example in Python using scipy.stats:
4. from scipy.stats import f\_oneway
5. # Assume `feature\_1\_cammeo` and `feature\_1\_osmancik` are the respective arrays for Cammeo and Osmancik varieties for a particular feature
6. stat, p\_value = f\_oneway(feature\_1\_cammeo, feature\_1\_osmancik)
7. if p\_value < 0.05:
8. print("Reject the null hypothesis, significant difference between groups")
9. else:
10. print("Fail to reject the null hypothesis, no significant difference")
11. **Interpreting Results:**
    * If the **p-value < 0.05**, you reject the null hypothesis, suggesting that the means of the rice varieties (Cammeo and Osmancik) are significantly different for that feature.
    * If the **p-value ≥ 0.05**, you fail to reject the null hypothesis, suggesting that there is no significant difference between the means of the two varieties for that feature.

**3. Summary of Statistical Analysis**

Here’s what you would conclude based on the results:

* **Normality Tests:** Determine if the features follow a normal distribution, which may influence your choice of subsequent statistical tests.
* **ANOVA Results:** Identify whether there are significant differences between the means of the rice varieties across the selected features. If you find significant differences, you can further investigate which features contribute most to these differences.

**Next Steps:**

* Conduct **pairwise comparisons** (e.g., using Tukey's HSD test) if you find significant differences in ANOVA and want to determine exactly which groups differ.
* Visualize the distributions of features (e.g., box plots, violin plots) to aid in the interpretation of the ANOVA results.

If you need more detailed code examples or further interpretation, feel free to ask!

Certainly! Below is a Python code that covers statistical regression, visualization, and ANOVA test for the **Rice Cammeo and Osmancik** dataset.

Before running the code, make sure you have installed the necessary libraries:

pip install pandas numpy scipy matplotlib seaborn

Now, let’s walk through the code:

1. **Data Loading**: We'll load the dataset from the UCI repository.
2. **Data Preprocessing**: We'll clean the dataset if needed and separate the features.
3. **ANOVA Test**: We'll perform a One-Way ANOVA test to check for significant differences between the Cammeo and Osmancik rice varieties.
4. **Visualization**: We'll create visualizations (histograms, boxplots) for data distribution and ANOVA results.
5. **Regression**: We will perform a simple linear regression to explore the relationship between features.

**Python Code:**

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

from scipy.stats import f\_oneway, shapiro

from sklearn.linear\_model import LinearRegression

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import mean\_squared\_error

# 1. Load the dataset

url = "https://archive.ics.uci.edu/ml/machine-learning-databases/545/rice%20cammeo%20and%20osmancik.csv"

df = pd.read\_csv(url)

# Check for the first few rows to understand the structure

print(df.head())

# 2. Data Preprocessing: If necessary, clean the dataset (handle missing values, encoding, etc.)

df = df.dropna() # Dropping missing values (if any)

# 3. Visualization - Histograms for data distribution

plt.figure(figsize=(10, 6))

df.hist(bins=20, figsize=(15, 10))

plt.suptitle('Histograms of Features', fontsize=16)

plt.show()

# 4. Perform Shapiro-Wilk test for normality (example for one feature)

stat, p\_value = shapiro(df['feature\_name']) # Replace 'feature\_name' with actual feature

if p\_value > 0.05:

print("Data follows normal distribution")

else:

print("Data does not follow normal distribution")

# 5. Boxplots to compare distributions between Cammeo and Osmancik

sns.boxplot(x='variety', y='feature\_name', data=df) # Replace 'variety' and 'feature\_name'

plt.title('Boxplot of feature\_name for Cammeo and Osmancik')

plt.show()

# 6. ANOVA Test

# Example: ANOVA for a specific feature between the two rice varieties (Cammeo and Osmancik)

cammeo\_data = df[df['variety'] == 'Cammeo']['feature\_name'] # Replace 'variety' and 'feature\_name'

osmancik\_data = df[df['variety'] == 'Osmancik']['feature\_name'] # Replace 'variety' and 'feature\_name'

# Perform One-Way ANOVA

stat, p\_value = f\_oneway(cammeo\_data, osmancik\_data)

if p\_value < 0.05:

print("Reject the null hypothesis: There is a significant difference in means")

else:

print("Fail to reject the null hypothesis: No significant difference in means")

# 7. Simple Linear Regression - Example (predicting a feature based on other features)

# Prepare data for regression (example using 'feature\_1' as predictor and 'feature\_2' as target)

X = df[['feature\_1']] # Predictor variable (replace 'feature\_1' with actual feature)

y = df['feature\_2'] # Target variable (replace 'feature\_2' with actual feature)

# Split data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Create and train the linear regression model

model = LinearRegression()

model.fit(X\_train, y\_train)

# Predict on test data

y\_pred = model.predict(X\_test)

# Evaluate the model

mse = mean\_squared\_error(y\_test, y\_pred)

print(f'Mean Squared Error: {mse}')

# Plotting regression results

plt.figure(figsize=(8, 6))

plt.scatter(X\_test, y\_test, color='blue', label='Actual data')

plt.plot(X\_test, y\_pred, color='red', linewidth=2, label='Regression line')

plt.title('Linear Regression: Feature\_1 vs Feature\_2')

plt.xlabel('Feature\_1')

plt.ylabel('Feature\_2')

plt.legend()

plt.show()

**Explanation of Each Section:**

**1. Data Loading:**

* We load the dataset from the URL provided.
* Use df.head() to check the first few rows and understand the structure.

**2. Data Preprocessing:**

* We drop any missing values with df.dropna() to ensure that we’re working with clean data.

**3. Visualization:**

* **Histograms**: These plots help visualize the distribution of each feature.
* **Boxplots**: We use seaborn.boxplot() to compare how the feature values are distributed between **Cammeo** and **Osmancik** varieties.

**4. Shapiro-Wilk Normality Test:**

* We test the normality of a specific feature with the **Shapiro-Wilk test** using scipy.stats.shapiro(). This will check if the feature is normally distributed.

**5. ANOVA Test:**

* We perform an **ANOVA test** using scipy.stats.f\_oneway() to check if there is a significant difference between the means of a specific feature for the two rice varieties. If the **p-value < 0.05**, it means there's a significant difference.

**6. Simple Linear Regression:**

* We build a **linear regression model** using sklearn.linear\_model.LinearRegression(). The model predicts one feature (e.g., feature\_2) based on another (e.g., feature\_1).
* We evaluate the model using **Mean Squared Error (MSE)** and visualize the regression line.

**Notes:**

* You need to replace feature\_name with actual feature names in the dataset (e.g., protein, moisture, etc.).
* If you want to perform multiple linear regression (using more than one predictor), simply adjust the X variable to include multiple features.
* The ANOVA test compares two groups (Cammeo vs. Osmancik), but if there are more groups, you can adjust the groups accordingly.

This code will help you conduct a comprehensive statistical analysis on the **Rice Cammeo and Osmancik** dataset. Let me know if you need further clarification or adjustments!

Certainly! Below is an example of a complete Python script for **data processing**, **normal distribution test**, **linear regression**, **multiple regression**, **visualization**, and **ANOVA test** for the dataset you mentioned: **Rice Cammeo and Osmancik** from UCI's Machine Learning Repository.

**Steps:**

1. **Data Processing**: Loading and cleaning the dataset.
2. **Normal Distribution Test**: Checking if the features follow a normal distribution.
3. **Linear Regression**: Fitting a linear regression model for one feature.
4. **Multiple Regression**: Fitting a multiple regression model using multiple features.
5. **ANOVA Test**: Comparing the means of features between Cammeo and Osmancik varieties.
6. **Visualization**: Visualizing histograms, boxplots, regression lines, and more.

Before running this code, ensure you have the necessary Python libraries installed by running:

pip install pandas numpy seaborn matplotlib scipy scikit-learn

**Full Python Example:**

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

from scipy.stats import f\_oneway, shapiro

from sklearn.linear\_model import LinearRegression

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import mean\_squared\_error

from sklearn.preprocessing import StandardScaler

# 1. Load the dataset

url = "https://archive.ics.uci.edu/ml/machine-learning-databases/545/rice%20cammeo%20and%20osmancik.csv"

df = pd.read\_csv(url)

# Check the first few rows to understand the dataset structure

print(df.head())

# 2. Data Preprocessing: Clean dataset (if needed)

# Drop any rows with missing values

df = df.dropna()

# We assume the dataset has columns like 'variety', 'feature1', 'feature2', etc.

# Replace 'feature1', 'feature2' with actual column names based on the dataset structure.

# 3. Visualize the distribution of features

plt.figure(figsize=(12, 6))

sns.histplot(df['feature\_1'], kde=True, bins=20)

plt.title('Distribution of Feature 1')

plt.show()

# Boxplot for Cammeo vs Osmancik comparison

sns.boxplot(x='variety', y='feature\_1', data=df)

plt.title('Boxplot of Feature\_1 by Rice Variety')

plt.show()

# 4. Normal Distribution Test (Shapiro-Wilk test)

stat, p\_value = shapiro(df['feature\_1']) # Replace with an actual feature name

print(f"Shapiro-Wilk Test for Feature\_1: stat={stat}, p-value={p\_value}")

if p\_value > 0.05:

print("Data for Feature\_1 follows a normal distribution.")

else:

print("Data for Feature\_1 does not follow a normal distribution.")

# 5. ANOVA Test (Comparing Cammeo and Osmancik for a particular feature)

cammeo\_data = df[df['variety'] == 'Cammeo']['feature\_1'] # Replace with actual column names

osmancik\_data = df[df['variety'] == 'Osmancik']['feature\_1'] # Replace with actual column names

# Perform One-Way ANOVA

stat, p\_value = f\_oneway(cammeo\_data, osmancik\_data)

print(f"ANOVA Test Result for Feature\_1: stat={stat}, p-value={p\_value}")

if p\_value < 0.05:

print("Reject the null hypothesis: There is a significant difference in means between Cammeo and Osmancik.")

else:

print("Fail to reject the null hypothesis: No significant difference in means.")

# 6. Linear Regression (Example with one feature)

X = df[['feature\_1']] # Predictor variable (replace with actual feature)

y = df['feature\_2'] # Target variable (replace with actual feature)

# Split data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Create and train the linear regression model

lin\_reg = LinearRegression()

lin\_reg.fit(X\_train, y\_train)

# Predict on test data

y\_pred = lin\_reg.predict(X\_test)

# Evaluate the model

mse = mean\_squared\_error(y\_test, y\_pred)

print(f"Linear Regression MSE: {mse}")

# Plotting regression results

plt.figure(figsize=(8, 6))

plt.scatter(X\_test, y\_test, color='blue', label='Actual data')

plt.plot(X\_test, y\_pred, color='red', linewidth=2, label='Regression line')

plt.title('Linear Regression: Feature\_1 vs Feature\_2')

plt.xlabel('Feature\_1')

plt.ylabel('Feature\_2')

plt.legend()

plt.show()

# 7. Multiple Linear Regression (Using multiple features)

# Let's use multiple features to predict 'feature\_2'

X\_multiple = df[['feature\_1', 'feature\_3', 'feature\_4']] # Replace with actual feature names

y\_multiple = df['feature\_2']

# Standardize the data (optional but often useful for multiple regression)

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X\_multiple)

# Split data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y\_multiple, test\_size=0.2, random\_state=42)

# Create and train the multiple regression model

multi\_reg = LinearRegression()

multi\_reg.fit(X\_train, y\_train)

# Predict on test data

y\_pred\_multi = multi\_reg.predict(X\_test)

# Evaluate the model

mse\_multi = mean\_squared\_error(y\_test, y\_pred\_multi)

print(f"Multiple Linear Regression MSE: {mse\_multi}")

# Plotting Multiple Regression predictions

plt.figure(figsize=(8, 6))

plt.scatter(y\_test, y\_pred\_multi, color='green')

plt.plot([y\_test.min(), y\_test.max()], [y\_test.min(), y\_test.max()], color='red', lw=2)

plt.title('Multiple Regression: Actual vs Predicted (Feature\_2)')

plt.xlabel('Actual')

plt.ylabel('Predicted')

plt.show()

# 8. Visualize the relationship between features (Pairplot)

sns.pairplot(df[['feature\_1', 'feature\_2', 'feature\_3']]) # Replace with actual feature names

plt.show()

**Step-by-Step Explanation:**

1. **Data Loading**:
   * We load the dataset directly from the UCI repository URL using pd.read\_csv().
   * We drop any rows with missing values using dropna() for simplicity.
2. **Visualizations**:
   * We visualize the distribution of **Feature 1** using a **histogram**.
   * A **boxplot** is used to visualize the distribution of **Feature 1** between the two rice varieties (Cammeo and Osmancik).
3. **Normal Distribution Test**:
   * The **Shapiro-Wilk test** (shapiro()) is used to test if **Feature 1** follows a normal distribution.
   * The result of this test is printed, and based on the **p-value**, we either accept or reject the null hypothesis (normal distribution).
4. **ANOVA Test**:
   * We conduct a **One-Way ANOVA** test using scipy.stats.f\_oneway() to compare the means of **Feature 1** between **Cammeo** and **Osmancik** rice varieties.
   * A **p-value** < 0.05 indicates a significant difference between the two varieties.
5. **Linear Regression**:
   * We use **simple linear regression** (LinearRegression()) to predict **Feature 2** based on **Feature 1**.
   * The model is trained, evaluated using **Mean Squared Error (MSE)**, and the regression line is plotted.
6. **Multiple Regression**:
   * We build a **multiple linear regression model** using **Feature 1**, **Feature 3**, and **Feature 4** to predict **Feature 2**.
   * The regression model is evaluated, and predictions are visualized.
7. **Visualization of Feature Relationships**:
   * We use seaborn.pairplot() to visualize the relationships between multiple features, helping understand how they correlate with each other.

**Notes:**

* Replace 'feature\_1', 'feature\_2', etc., with the actual feature names in the dataset.
* The code assumes that the dataset has a column named 'variety' that indicates whether the rice is **Cammeo** or **Osmancik**. Adjust column names based on the actual dataset structure.
* The **multiple regression** step uses standardization (StandardScaler()) to improve the model performance, especially when working with features of different scales.

This code provides a comprehensive pipeline for data analysis, statistical testing, and machine learning tasks on the **Rice Cammeo and Osmancik** dataset. Let me know if you need further assistance!