```
# Import necessary libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.cluster import KMeans
from sklearn.metrics import mean_squared_error, r2_score

# Load the dataset
url = 'https://archive.ics.uci.edu/static/public/913/data.csv'
data = pd.read_csv(url)

# Display first few rows of the dataset and the columns
print("First few rows of the dataset:")
print(data.head())
print("\nDataset columns:")
print(data.columns)

# Check for missing values
print("\nMissing values in each column:")
print(data.isnull().sum())

# Drop missing values or impute them
data.dropna(inplace=True)

# Set the target variable to the correct column name
target_variable = 'GY'  # Correct target variable

# Check if the target variable exists
if target_variable not in data.columns:
    print(f"\n'{target_variable}' column not found in the dataset. Please check the column names.")
else:
    # Identify categorical and numeric columns
    categorical_cols = data.select_dtypes(include=['object']).columns.tolist()
    numeric_cols = data.select_dtypes(include=[np.number]).columns.tolist()

    # Remove target variable from numeric columns if it exists
    numeric_cols = [col for col in numeric_cols if col != target_variable]

    # Create a preprocessor
    preprocessor = ColumnTransformer(
        transformers=[
            ('num', StandardScaler(), numeric_cols),
            ('cat', OneHotEncoder(), categorical_cols)
        ]
    )

    # Split the data into features (X) and target (y)
    X = data.drop(columns=[target_variable])  # Drop target variable
    y = data[target_variable]

    # Transform the features using the preprocessor
    X_transformed = preprocessor.fit_transform(X)

    # Split the data into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(X_transformed, y, test_size=0.2, random_state=42)

    # Train Linear Regression model
    lr_model = LinearRegression()
    lr_model.fit(X_train, y_train)

    # Predict and evaluate Linear Regression model
    y_pred_lr = lr_model.predict(X_test)
    mse_lr = mean_squared_error(y_test, y_pred_lr)
    r2_lr = r2_score(y_test, y_pred_lr)

    print(f"\nLinear Regression MSE: {mse_lr}")
    print(f"Linear Regression R^2: {r2_lr}")

    # Train Decision Tree Regressor
    dt_model = DecisionTreeRegressor(random_state=42)
    dt_model.fit(X_train, y_train)

    # Predict and evaluate Decision Tree Regressor
    y_pred_dt = dt_model.predict(X_test)
```

```
mse_dt = mean_squared_error(y_test, y_pred_dt)
r2_dt = r2_score(y_test, y_pred_dt)

print(f"\nDecision Tree Regressor MSE: {mse_dt}")
print(f"Decision Tree Regressor R^2: {r2_dt}")

# Train Random Forest Regressor
rf_model = RandomForestRegressor(random_state=42)
rf_model.fit(X_train, y_train)

# Predict and evaluate Random Forest Regressor
y_pred_rf = rf_model.predict(X_test)
mse_rf = mean_squared_error(y_test, y_pred_rf)
r2_rf = r2_score(y_test, y_pred_rf)

print(f"\nRandom Forest Regressor MSE: {mse_rf}")
print(f"Random Forest Regressor R^2: {r2_rf}")

# K-Means Clustering using the transformed features
kmeans = KMeans(n_clusters=3, random_state=42)  # Example with 3 clusters
clusters = kmeans.fit_predict(X_transformed)  # Use the transformed features for clustering
data['Cluster'] = clusters  # Add cluster labels to the original data

# Visualize the clusters
plt.figure(figsize=(10, 5))
if 'PH' in data.columns:  # Example visualization using PH
    sns.scatterplot(x=data['PH'], y=data[target_variable], hue=data['Cluster'], palette='viridis')
    plt.title('K-Means Clustering of Soybean Cultivars')
    plt.xlabel('PH')
    plt.ylabel('Grain Yield (GY)')
    plt.show()
else:
    print("\n'PH' column not found.")
```

```
First few rows of the dataset:
   Season    Cultivar  Repetition     PH    IFP    NLP     NGP   NGL   NS  \
0       1  NEO 760 CE           1  58.80  15.20   98.2  177.80  1.81  5.2
1       1  NEO 760 CE           2  58.60  13.40  102.0  195.00  1.85  7.2
2       1  NEO 760 CE           3  63.40  17.20  100.4  203.00  2.02  6.8
3       1  NEO 760 CE           4  60.27  15.27  100.2  191.93  1.89  6.4
4       1   MANU IPRO           1  81.20  18.00   98.8  173.00  1.75  7.4

      MHG       GY
0  152.20  3232.82
1  141.69  3517.36
2  148.81  3391.46
3  148.50  3312.58
4  145.59  3230.99

Dataset columns:
Index(['Season', 'Cultivar', 'Repetition', 'PH', 'IFP', 'NLP', 'NGP', 'NGL',
       'NS', 'MHG', 'GY'],
      dtype='object')

Missing values in each column:
Season        0
Cultivar      0
Repetition    0
PH            0
IFP           0
NLP           0
NGP           0
NGL           0
NS            0
MHG           0
GY            0
dtype: int64

Linear Regression MSE: 123571.9552071181
Linear Regression R^2: 0.5227462242215724

Decision Tree Regressor MSE: 164997.0915909664
Decision Tree Regressor R^2: 0.3627560167494073

Random Forest Regressor MSE: 134788.30640879794
Random Forest Regressor R^2: 0.4794269617522977
```