```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.simplefilter("ignore")
```

```
pip install ucimlrepo
```

```
Requirement already satisfied: ucimlrepo in /usr/local/lib/python3.10/dist-packages (0.0.3)
```

```
from ucimlrepo import fetch_ucirepo

# fetch dataset
iris = fetch_ucirepo(id=53)

# data (as pandas dataframes)
X = iris.data.features
y = iris.data.targets

# metadata
print(iris.metadata)

# variable information
print(iris.variables)
```

```
53, 'name': 'Iris', 'repository_url': 'https://archive.ics.uci.edu/dataset/53/iris', 'data_url': 'https://archive.ics.uci.edu/static
name     role         type demographic  \
ngth  Feature   Continuous        None
idth  Feature   Continuous        None
ngth  Feature   Continuous        None
idth  Feature   Continuous        None
lass   Target  Categorical        None

                            description units missing_values
                                   None    cm             no
                                   None    cm             no
                                   None    cm             no
                                   None    cm             no
iris plant: Iris Setosa, Iris Versico...  None             no
```

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

```
# Load the Iris dataset
data = load_iris()
X, y = data.data, data.target
```

```
# Split the dataset 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)
```

```
# Standardize the feature values
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
# Train a Logistic Regression model
clf = LogisticRegression()
clf.fit(X_train, y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
# Make predictions on the test data
y_pred = clf.predict(X_test)
```

```
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
```

```
Accuracy: 1.0
```