📝 Project Report: Machine Learning Data Preprocessing on Iris and Wine Datasets

📌 1. Introduction

In this project, we explore **data preprocessing techniques** using two well-known datasets: **Iris** and **Wine**, both available in `sklearn.datasets`. Preprocessing is a critical step in machine learning to ensure data quality, model performance, and accurate predictions.

📚 2. Dataset Description

☑️ *Iris Dataset*

- 150 samples, 4 features
- Features: sepal length, sepal width, petal length, petal width
- Target: 3 classes (setosa, versicolor, virginica)

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

☑️ *Wine Dataset*

- 178 samples, 13 features
- Features: chemical analysis attributes of wine cultivars
- Target: 3 classes (class_0, class_1, class_2)

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hu
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28	2.29	5.64	1.0
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26	1.28	4.38	1.0
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.30	2.81	5.68	1.0
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.24	2.18	7.80	0.8
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.39	1.82	4.32	1.0

🛠️ 3. Preprocessing Steps

◇ *Step 1: Loading Data and importing libraries*

Both datasets are loaded using ***`load_iris()`*** and ***`load_wine()`*** functions.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.datasets import load_iris, load_wine
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import LabelEncoder, StandardScaler, KBinsDiscretizer
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.decomposition import PCA
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
from scipy import stats
```

◇ *Step 2: Missing Value Handling*

To simulate real-world scenarios, we inserted artificial missing values and used ***`SimpleImputer`*** (mean strategy) to fill them.

```
imputer = SimpleImputer(strategy='mean')
df.iloc[:, :-1] = imputer.fit_transform(df.iloc[:, :-1])
```

Step 3: Label Encoding

Although both datasets have numeric labels, ***LabelEncoder*** was applied for demonstration.

```
le = LabelEncoder()
df['target'] = le.fit_transform(df['target'])
```

Step 4: Feature Scaling

We used ***StandardScaler*** to standardize the dataset.

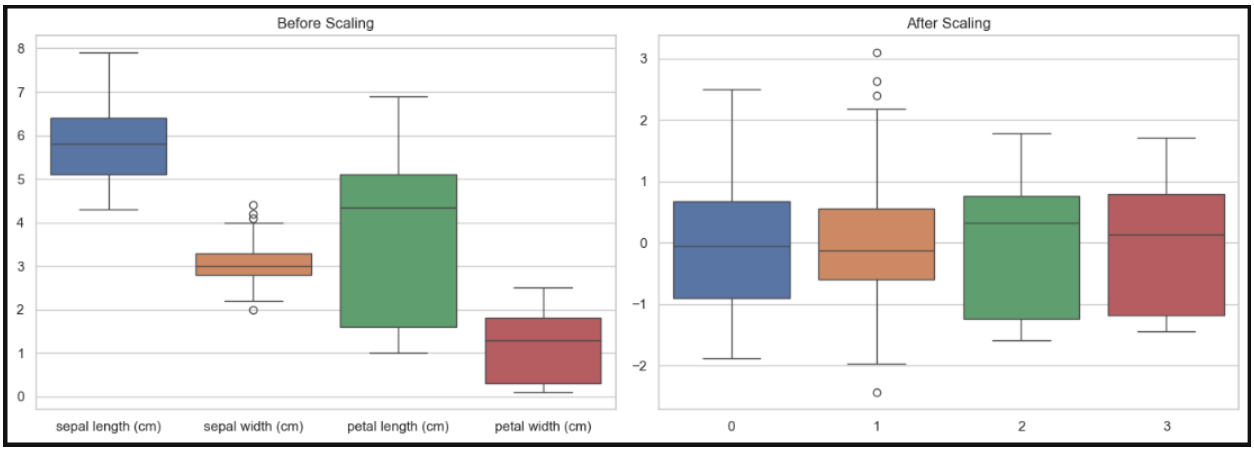
```
scaler = StandardScaler()
X = df.drop(columns='target')
y = df['target']

X_scaled = scaler.fit_transform(X)
```

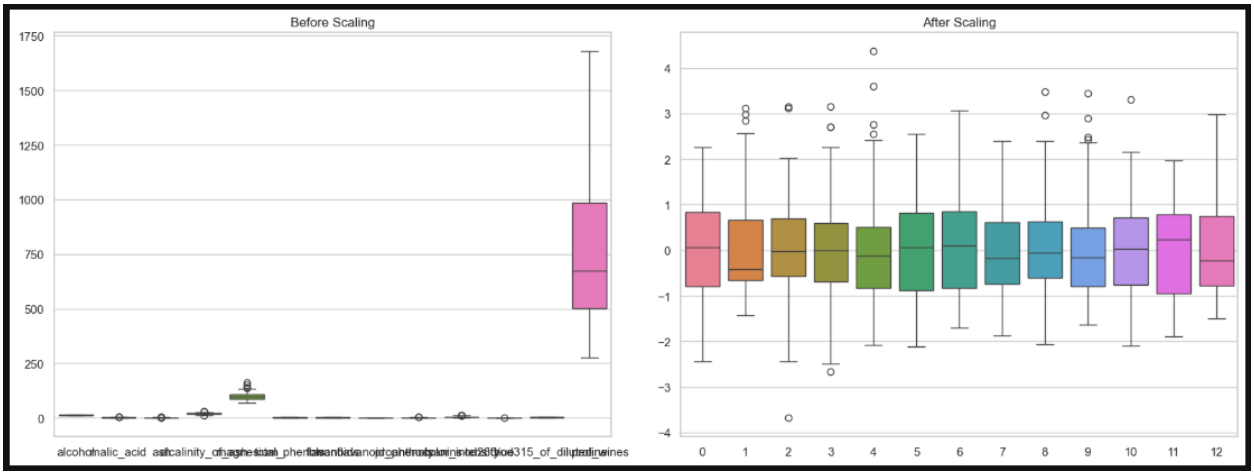
Step 5: Outlier Detection

Boxplots were plotted **before and after scaling** to visually identify outliers.

IRIS :



WINE :



4. Feature Engineering

Feature Selection

Using ***SelectKBest*** and ANOVA F-test to select the most relevant features.

```
selector = SelectKBest(score_func=f_classif, k=4)
X_selected = selector.fit_transform(X_scaled, y)
```

Discretization

We converted continuous values into bins using ***KBinsDiscretizer***.

```
binner = KBinsDiscretizer(n_bins=3, encode='ordinal', strategy='quantile')
X_binned = binner.fit_transform(X_scaled)
```

IRIS :

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	1.0	2.0	0.0	0.0	0
1	0.0	1.0	0.0	0.0	0
2	0.0	2.0	1.0	0.0	0
3	0.0	1.0	0.0	0.0	0
4	0.0	2.0	0.0	0.0	0

WINE :

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue
0	2.0	1.0	1.0	0.0	2.0	2.0	2.0	0.0	2.0	1.0	1.0
1	1.0	1.0	0.0	0.0	1.0	2.0	2.0	0.0	0.0	1.0	1.0
2	1.0	1.0	2.0	1.0	1.0	2.0	2.0	1.0	2.0	2.0	1.0
3	2.0	1.0	2.0	0.0	2.0	2.0	2.0	0.0	2.0	2.0	0.0
4	1.0	2.0	2.0	2.0	2.0	1.0	2.0	1.0	1.0	1.0	1.0

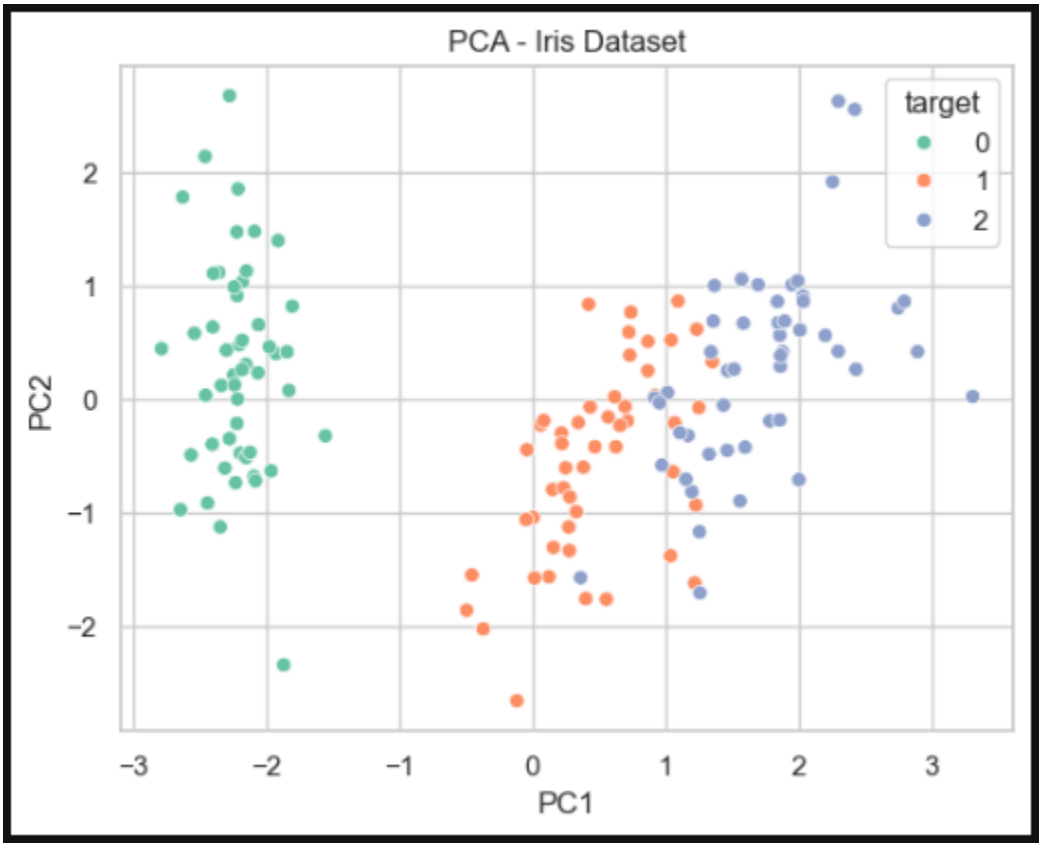
5. Dimensionality Reduction

Principal Component Analysis (PCA)

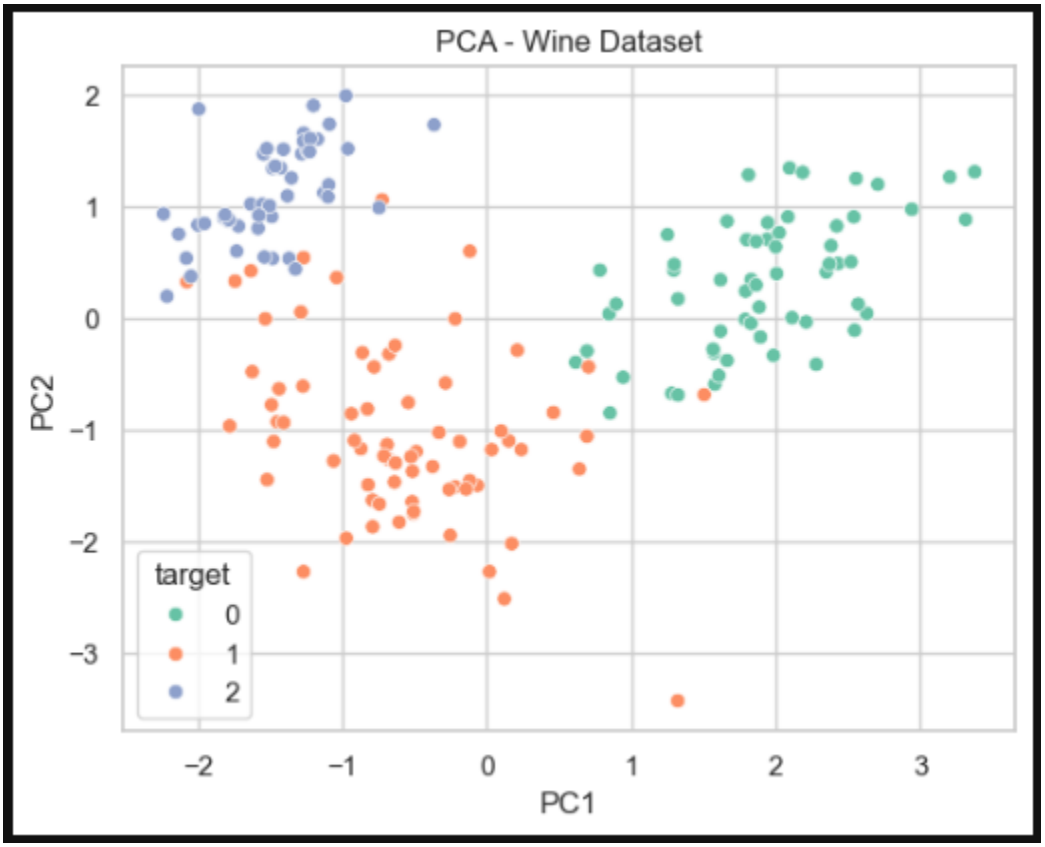
PCA was applied to reduce the dataset to 2D for visualization.

```
X_pca_input = selector.fit_transform(X_scaled, y)
pca = PCA(n_components=2)
pca_components = pca.fit_transform(X_pca_input)
```

IRIS:



WINE:

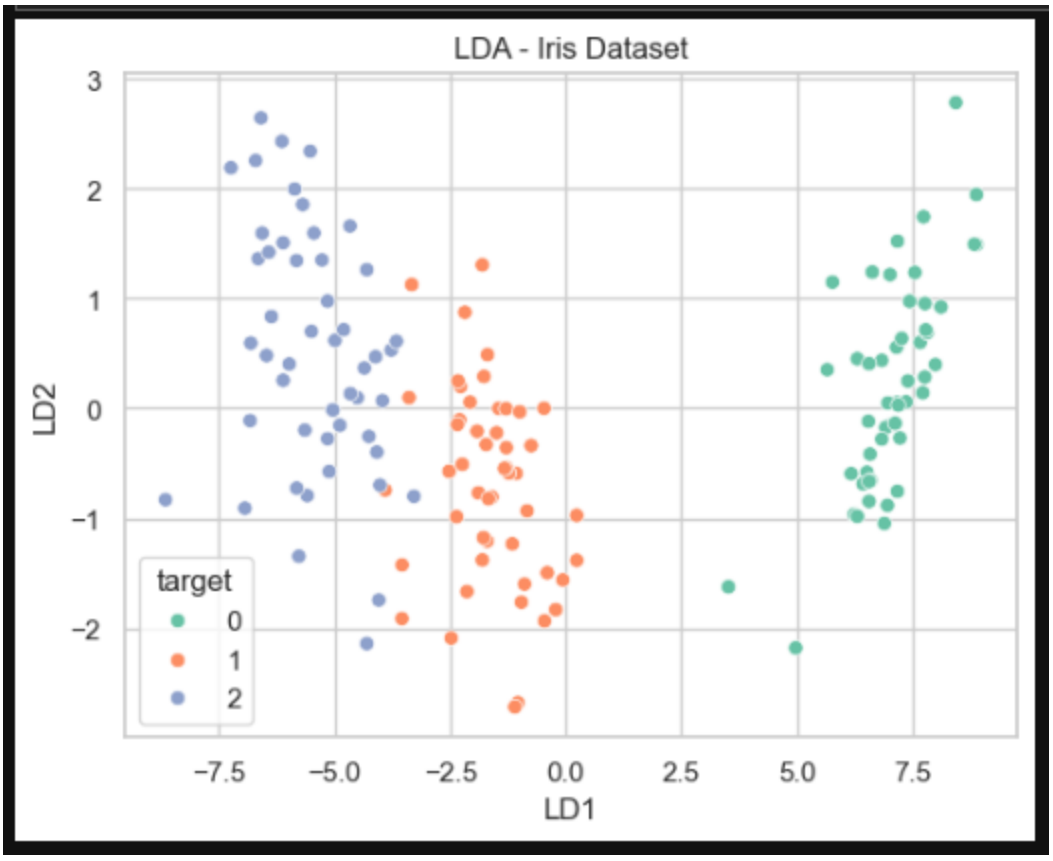


- ***Linear Discriminant Analysis (LDA)***

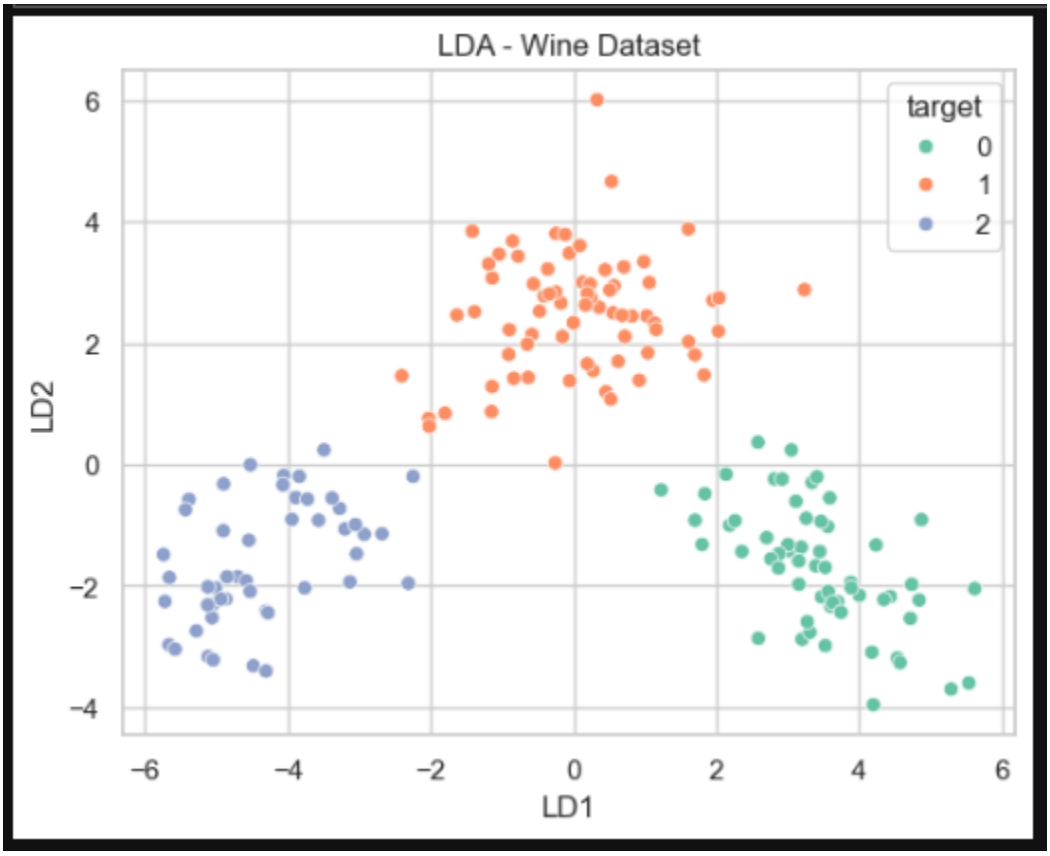
LDA was used to maximize class separation.

```
lda = LDA(n_components=2)
lda_components = lda.fit_transform(X_scaled, y)
```

IRIS:


LDA - Iris Dataset
LD2
LD1
target
0
1
2
3
2
1
0
−1
−2
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5

WINE:



6. Model Training and Evaluation

A simple **Logistic Regression** model was trained using the selected features and evaluated using classification report.

```
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

print("\n🔹 Classification Report:")
print(classification_report(y_test, y_pred))
```

Classification report includes precision, recall, f1-score, and accuracy for each class.

☑ 7. Results Summary

Dataset	Accuracy	Key Selected Features	PCA Visual Clarity	LDA Visual Clari
Iris	High	Petal Length, Petal Width	High	High
Wine	High	Flavanoids, OD280/OD315 & others	Moderate	High

📌 Environment: Jupyter Notebook

File link: Mohd-Faizan-Rashid/python-ml-data-prerocessing-project

The aim of this project is to demonstrate **all data preprocessing techniques** used in Machine Learning workflows. We had use **two classical datasets** — Iris and Wine — to apply and visualize a variety of preprocessing steps.

Thank you

📓 Submitted by: Mohd Faizan Rashid