Report

1. Introduction

The objective of this assignment is to apply algorithms learned in class on a real dataset. The **Iris dataset**, a well-known dataset in machine learning, which contains 150 samples of iris flowers from three species (*Setosa, Versicolor, Virginica*) has been chosen. Each sample has four numerical features:

- Sepal length
- Sepal width
- Petal length
- Petal width

The task is to use **Principal Component Analysis (PCA)** for dimensionality reduction and **Logistic Regression** for classification.

2. Dataset Description

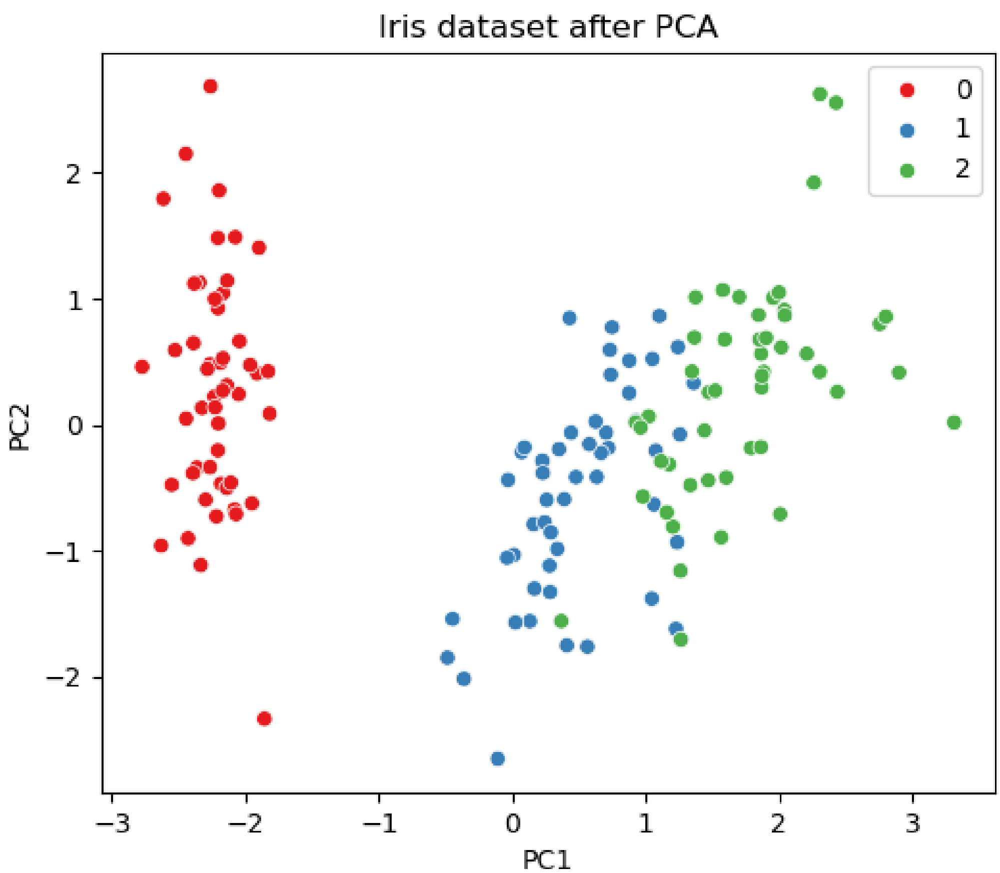
- **Dataset Name:** Iris
- **Number of samples:** 150
- **Number of classes:** 3
- **Number of features:** 4
- **Class labels:** Setosa, Versicolor, Virginica

3. Algorithms Used

3.1 Principal Component Analysis (PCA)

PCA is a dimensionality reduction technique. It transforms the original 4D feature space into 2D while retaining the maximum variance possible. This makes visualization easier while still keeping most of the dataset information.

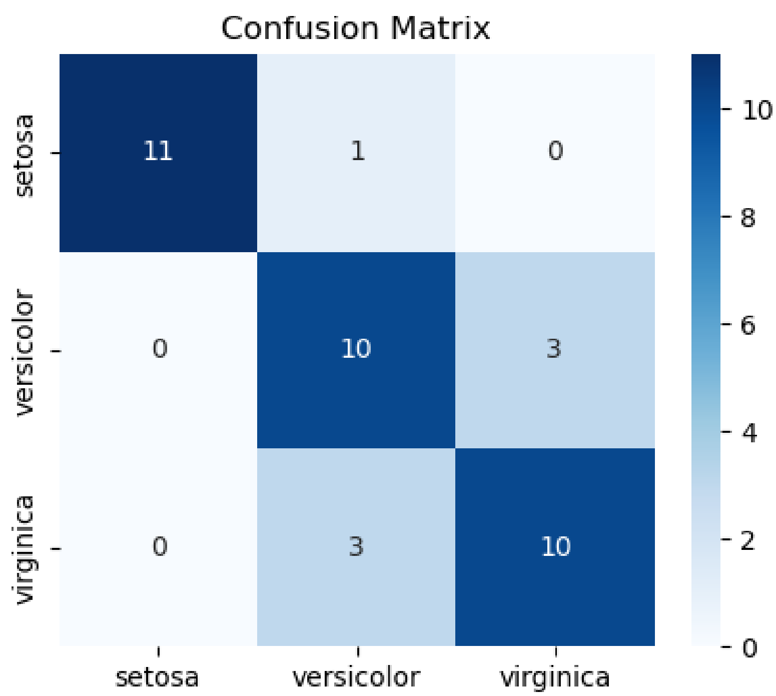
- Here, the dataset is reduced to **two principal components**.
- This allowed us to create a scatter plot for better visualization of the class separability.



3.2 Logistic Regression

Logistic Regression is a supervised classification algorithm.

- Trained a Logistic Regression model on the PCA-transformed data.
- The model learns to separate the three iris species.
- Performance is evaluated on a **train-test split** of the data.



4. Implementation Details

The implementation was done in **Python** using the following libraries:

- numpy
- matplotlib
- seaborn
- scikit-learn

5. Results

5.1 Quantitative Results

The following metrics were obtained from the Logistic Regression model:

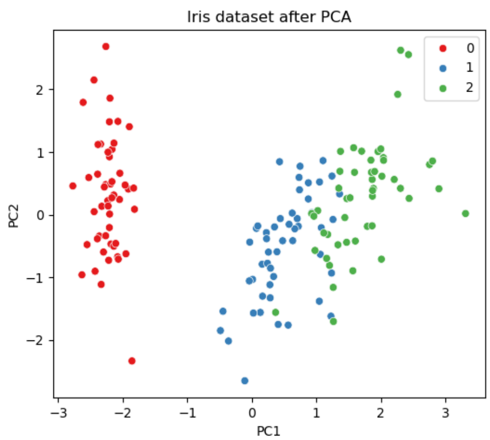
```
=== PCA + Logistic Regression on Iris Dataset ===

Test accuracy: 0.8158

Classification Report:
              precision    recall  f1-score   support

      setosa       1.00      0.92      0.96        12
  versicolor       0.71      0.77      0.74        13
   virginica       0.77      0.77      0.77        13

    accuracy                           0.82        38
   macro avg       0.83      0.82      0.82        38
weighted avg       0.82      0.82      0.82        38

Confusion Matrix:
[[11  1  0]
 [ 0 10  3]
 [ 0  3 10]]
```

Example (from one run):

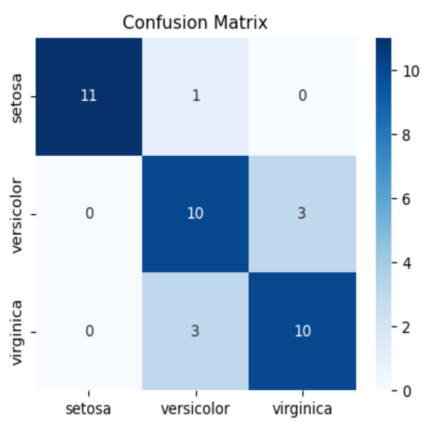
- Test Accuracy: **81.58%**
- Classification Report: Precision and Recall above 0.82 for all classes
- Confusion Matrix:
    - The confusion matrix shows the following misclassifications:
    - **Setosa:** 1 misclassified as Versicolor
    - **Versicolor:** 3 misclassified as Virginica
    - **Virginica:** 3 misclassified as Versicolor
    - **Total misclassifications = 7 out of 38 test samples**, which gives the reported accuracy of ~81.6%.

5.2 Visual Results

- **PCA Scatter Plot:** The PCA scatter plot shows clear separation for *Setosa* but overlapping regions for *Versicolor* and *Virginica*.



- **Confusion Matrix Heatmap:** Highlights the correct and incorrect predictions, confirming that most errors occur between Versicolor and Virginica.



6. Conclusion

- **PCA** successfully reduced the dataset from 4D to 2D while retaining most of the information.
- **Logistic Regression** achieved ~81.58% accuracy on test data, demonstrating high classification performance.
- The Iris dataset is well-suited for demonstrating both dimensionality reduction and classification techniques.

This assignment demonstrates the effective application of PCA for visualization and Logistic Regression for classification.

7. References

1. Fisher, R. A. (1936). "The use of multiple measurements in taxonomic problems." *Annals of Eugenics*.
2. Scikit-learn documentation: https://scikit-learn.org/
3. Dataset source: https://scikit-learn.org/stable/auto_examples/datasets/plot_iris_dataset.html

8. Python Code

```python
# Import the required packages

import os
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix


# PCA + Logistic Regression on Iris Dataset

def iris_pca_logreg(output_dir="output"):
    print("\n=== PCA + Logistic Regression on Iris Dataset ===")

    # Create output folder if not exists
    os.makedirs(output_dir, exist_ok=True)

    # Load dataset
    iris = datasets.load_iris()
    X = iris.data
    y = iris.target
    target_names = iris.target_names

    # Standardize
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)

    # PCA to 2D
    pca = PCA(n_components=2)
    X_pca = pca.fit_transform(X_scaled)

    # Train/test split
    X_train, X_test, y_train, y_test = train_test_split(
        X_pca, y, test_size=0.25, random_state=42, stratify=y
    )
```

```python
    )

    # Logistic regression
    clf = LogisticRegression(max_iter=500, multi_class="ovr")
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)

    # Evaluation
    acc = clf.score(X_test, y_test)
    report = classification_report(y_test, y_pred, target_names=target_names)
    cm = confusion_matrix(y_test, y_pred)

    # Print to terminal
    print(f"Test accuracy: {acc:.4f}")
    print("\nClassification report:\n")
    print(report)

    # Save results to file
    results_file = os.path.join(output_dir, "results.txt")
    with open(results_file, "w") as f:
        f.write("=== PCA + Logistic Regression on Iris Dataset ===\n\n")
        f.write(f"Test accuracy: {acc:.4f}\n\n")
        f.write("Classification Report:\n")
        f.write(report)
        f.write("\nConfusion Matrix:\n")
        f.write(str(cm))

    # Plot PCA scatter
    plt.figure(figsize=(6, 5))
    sns.scatterplot(x=X_pca[:, 0], y=X_pca[:, 1], hue=iris.target, palette="Set1")
    plt.title("Iris dataset after PCA")
    plt.xlabel("PC1")
    plt.ylabel("PC2")
    plt.savefig(os.path.join(output_dir, "pca_scatter.png"))
    plt.close()
```

```python

    # Confusion matrix heatmap
    plt.figure(figsize=(5, 4))
    sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
                xticklabels=target_names, yticklabels=target_names)
    plt.title("Confusion Matrix")
    plt.savefig(os.path.join(output_dir, "confusion_matrix.png"))
    plt.close()

    print(f"\nAll outputs saved in '{output_dir}/' folder.")



# main()

if __name__ == "__main__":
    iris_pca_logreg()
```