Classification of Dry Beans using SVM and PCA

Name – Divyam Gupta

Roll No. – 2201AI48

1. Introduction

In this project, the **Dry Bean dataset** was used to classify bean types using a **Support Vector Machine (SVM) with Radial Basis Function (RBF) kernel**. To remove redundancy, improve computational efficiency and make data easier to visualize **Principal Component Analysis (PCA)** was applied.

2. Dataset

The dataset used is the **Dry Bean dataset** from the UCI Machine Learning Repository (link: https://www.kaggle.com/datasets/sansuthi/dry-bean-dataset?resource=download&select=Dry_Bean.csv). It consists of **13,611 instances** belonging to 7 different bean classes:

- SEKER
- BARBUNYA
- BOMBAY
- CALI
- HOROZ
- SIRA
- DERMASON

Each instance is described by 16 numerical features that represent shape and texture properties of the beans (e.g., area, perimeter, axis lengths, eccentricity, compactness). The target variable is categorical (class label) indicating the bean type.

3. Methods

The methodology followed consists of:

1. **Preprocessing**
    - Missing values were checked and none were found in the dataset.

    - Features were standardized using **z-score normalization** to bring them to a common scale, which is essential for both PCA and SVM.

2. **Dimensionality Reduction (PCA)**
    - PCA was applied to the 16 standardized features.
    - It was fitted on the scaled train data to retain 99% of the variance.
    - The first two principal components were used for 2D visualization.
3. **Classification (SVM with RBF Kernel)**
    - The dataset was split into training (80%) and testing (20%) sets.
    - An **SVM with RBF kernel** was trained to capture non-linear decision boundaries.
4. **Visualization**
    - PCA-reduced 2D representation of the data was plotted, with points colored according to their class labels.
    - The SVM decision boundaries were visualized in the 2D PCA space.

4. Implementation

- **Programming Language:** Python
- **Libraries Used:** NumPy, Pandas, Scikit-learn, Matplotlib, Seaborn

Steps Implemented:

1. Loaded the Dry Bean dataset and separated features (X) and target labels (y).
2. Split the dataset into train-test sets.
3. Applied StandardScaler to standardize features.
4. Performed PCA for visualization.
5. Trained an SVC(kernel='rbf') model with hyperparameters: C=10, gamma="scale".
6. Evaluated model accuracy and generated classification report.
7. Visualised the classification by using a confusion matrix.
8. Visualized the distribution of the dataset and the SVM boundary using Matplotlib

5. Code Snippets

```
pca = PCA(n_components=0.99, svd_solver='full')
X_train_pca = pca.fit_transform(X_train_scaled)
X_test_pca = pca.transform(X_test_scaled)

print(f"Original feature count: {X.shape[1]}")
print(f"Reduced feature count after PCA: {X_train_pca.shape[1]}")
```

```
pca2 = PCA(n_components=2)
X_pca_2d = pca2.fit_transform(X_train_scaled)
X_pca_2d_test = pca.transform(X_test_scaled)
plt.figure(figsize=(8,6))
sns.scatterplot(x=X_pca_2d[:,0], y=X_pca_2d[:,1], hue=y_train, palette="tab10", alpha=0.7)
plt.title("Dry Bean PCA Projection (2D)")
plt.xlabel("PC1")
plt.ylabel("PC2")
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```

PCA on the dataset

```
svm_rbf = SVC(kernel="rbf", C=10, gamma="scale", random_state=42)
svm_rbf.fit(X_train_pca, y_train)
```

Fitting SVM on the Data

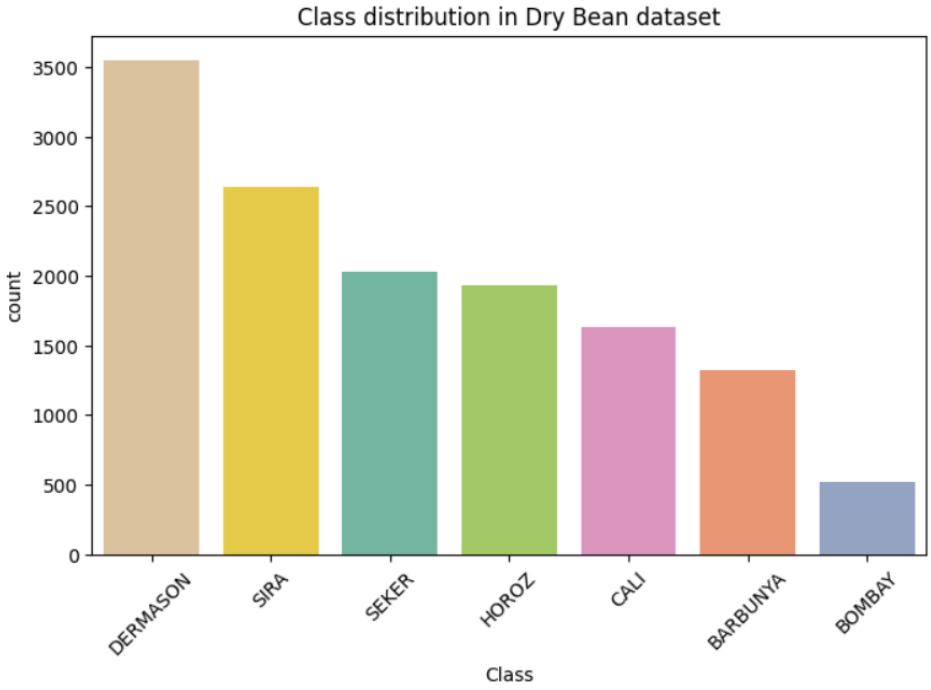
```
from sklearn.preprocessing import LabelEncoder

# Encode class labels to integers for visualization)
le = LabelEncoder()
y_train_enc = le.fit_transform(y_train)

svm_vis = SVC(kernel="rbf", C=10, gamma="scale")
svm_vis.fit(X_pca_2d, y_train_enc)

# Meshgrid
x_min, x_max = X_pca_2d[:,0].min() - 1, X_pca_2d[:,0].max() + 1
y_min, y_max = X_pca_2d[:,1].min() - 1, X_pca_2d[:,1].max() + 1
xx, yy = np.meshgrid(np.linspace(x_min, x_max, 300),
                     np.linspace(y_min, y_max, 300))

Z = svm_vis.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

plt.figure(figsize=(10,7))
plt.contourf(xx, yy, Z, alpha=0.3, cmap="tab10")

sns.scatterplot(x=X_pca_2d[:,0], y=X_pca_2d[:,1], hue=y_train, palette="tab10", edgecolor="k")
plt.title("SVM Decision Boundaries (on 2D PCA projection)")
plt.xlabel("PC1")
plt.ylabel("PC2")
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```

Visualising the data and SVM's decision boundary

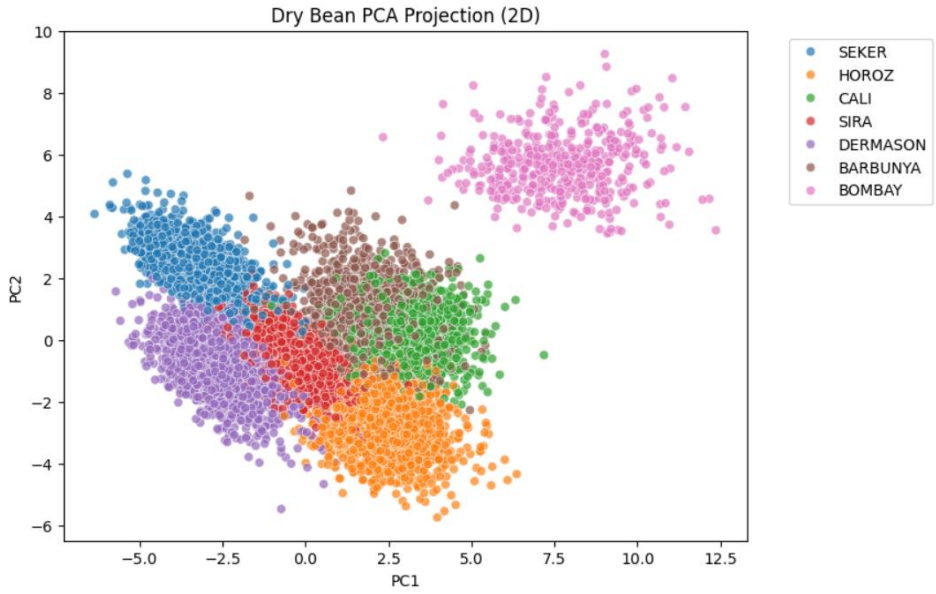
6. Results

- **PCA Analysis:**
    - PCA led to decrease of number of features from 16 to 7 while retaining 99% of the variation.
- **SVM Classification:**
    - The SVM with RBF kernel achieved **high accuracy of 92.45%** on the test set.
    - The following were the precision and recall (both weighted and macro):
        - `Macro Precision: 0.9373706756116549`
        - `Macro Recall: 0.935109902540867`
        - `Weighted Precision: 0.9243542101695529`
        - `Weighted Recall: 0.9243481454278369`
    - Confusion matrix showed strong performance across most classes, with minor misclassifications between similar beans (mostly DERMASON vs SIRA).

Here are the outputs –



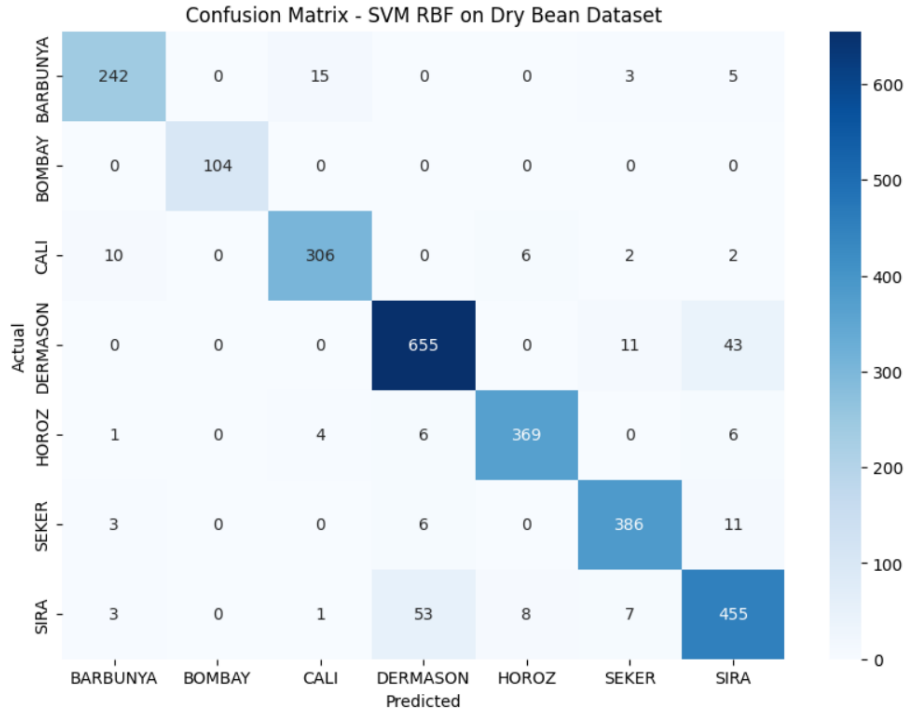
Data distribution among the classes



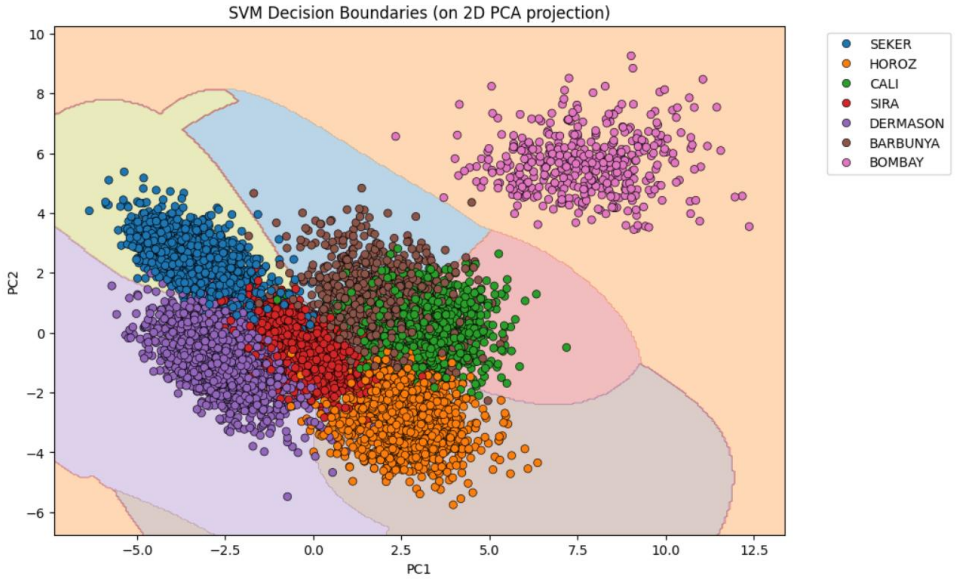
Visualisation of the data distribution

```
Accuracy: 0.9243481454278369

Classification Report:
              precision    recall  f1-score   support

    BARBUNYA       0.93      0.91      0.92       265
      BOMBAY       1.00      1.00      1.00       104
        CALI       0.94      0.94      0.94       326
    DERMASON       0.91      0.92      0.92       709
      HOROZ        0.96      0.96      0.96       386
      SEKER        0.94      0.95      0.95       406
        SIRA       0.87      0.86      0.87       527

    accuracy                           0.92      2723
   macro avg       0.94      0.94      0.94      2723
weighted avg       0.92      0.92      0.92      2723
```

Accuracy and Classification Report



Confusion Matrix



Visualisation of the SVM decision boundaries

7. Conclusion

This exercise demonstrated that **SVM with RBF kernel** is highly effective in classifying the Dry Bean dataset. PCA provided helped in reducing dimensionality and enabling visualization, though some class overlap persisted due to intrinsic feature similarities.