Use-Case-3

December 1, 2024

1 Module 8: Dimensionality Reduction

1.1 Case Study – 3

1.1.1 Dimensionality Reduction and Supervised Learning for Breast Cancer Classification: A Comparative Study of PCA, LDA, and Ensemble Methods.

[8]:
```python
# Step 0: Import required libraries
# Load and Explore the Data
import pandas as pd
import numpy as np
# 1. Load the digits dataset breast-cancer-data.csv
dataset=pd.read_csv('breast-cancer-data.csv')
dataset.info()
print(dataset.isnull().sum())
dataset.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 32 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   id                       569 non-null    int64
 1   diagnosis                569 non-null    object
 2   radius_mean              569 non-null    float64
 3   texture_mean             569 non-null    float64
 4   perimeter_mean           569 non-null    float64
 5   area_mean                569 non-null    float64
 6   smoothness_mean          569 non-null    float64
 7   compactness_mean         569 non-null    float64
 8   concavity_mean           569 non-null    float64
 9   concave points_mean      569 non-null    float64
 10  symmetry_mean            569 non-null    float64
 11  fractal_dimension_mean   569 non-null    float64
 12  radius_se                569 non-null    float64
 13  texture_se               569 non-null    float64
 14  perimeter_se             569 non-null    float64
 15  area_se                  569 non-null    float64
 16  smoothness_se            569 non-null    float64
```

```
 17  compactness_se           569 non-null    float64
 18  concavity_se             569 non-null    float64
 19  concave points_se        569 non-null    float64
 20  symmetry_se              569 non-null    float64
 21  fractal_dimension_se     569 non-null    float64
 22  radius_worst             569 non-null    float64
 23  texture_worst            569 non-null    float64
 24  perimeter_worst          569 non-null    float64
 25  area_worst               569 non-null    float64
 26  smoothness_worst         569 non-null    float64
 27  compactness_worst        569 non-null    float64
 28  concavity_worst          569 non-null    float64
 29  concave points_worst     569 non-null    float64
 30  symmetry_worst           569 non-null    float64
 31  fractal_dimension_worst  569 non-null    float64
dtypes: float64(30), int64(1), object(1)
memory usage: 142.4+ KB
id                          0
diagnosis                   0
radius_mean                 0
texture_mean                0
perimeter_mean              0
area_mean                   0
smoothness_mean             0
compactness_mean            0
concavity_mean              0
concave points_mean         0
symmetry_mean               0
fractal_dimension_mean      0
radius_se                   0
texture_se                  0
perimeter_se                0
area_se                     0
smoothness_se               0
compactness_se              0
concavity_se                0
concave points_se           0
symmetry_se                 0
fractal_dimension_se        0
radius_worst                0
texture_worst               0
perimeter_worst             0
area_worst                  0
smoothness_worst            0
compactness_worst           0
concavity_worst             0
concave points_worst        0
symmetry_worst              0
```

```
fractal_dimension_worst    0
dtype: int64
```

[8]:

```
         id diagnosis  radius_mean  texture_mean  perimeter_mean  area_mean  \
0    842302         M        17.99         10.38          122.80     1001.0
1    842517         M        20.57         17.77          132.90     1326.0
2  84300903         M        19.69         21.25          130.00     1203.0
3  84348301         M        11.42         20.38           77.58      386.1
4  84358402         M        20.29         14.34          135.10     1297.0

   smoothness_mean  compactness_mean  concavity_mean  concave points_mean  \
0          0.11840           0.27760          0.3001              0.14710
1          0.08474           0.07864          0.0869              0.07017
2          0.10960           0.15990          0.1974              0.12790
3          0.14250           0.28390          0.2414              0.10520
4          0.10030           0.13280          0.1980              0.10430

   …  radius_worst  texture_worst  perimeter_worst  area_worst  \
0  …         25.38          17.33           184.60      2019.0
1  …         24.99          23.41           158.80      1956.0
2  …         23.57          25.53           152.50      1709.0
3  …         14.91          26.50            98.87       567.7
4  …         22.54          16.67           152.20      1575.0

   smoothness_worst  compactness_worst  concavity_worst  concave points_worst  \
0            0.1622             0.6656           0.7119                0.2654
1            0.1238             0.1866           0.2416                0.1860
2            0.1444             0.4245           0.4504                0.2430
3            0.2098             0.8663           0.6869                0.2575
4            0.1374             0.2050           0.4000                0.1625

   symmetry_worst  fractal_dimension_worst
0          0.4601                  0.11890
1          0.2750                  0.08902
2          0.3613                  0.08758
3          0.6638                  0.17300
4          0.2364                  0.07678

[5 rows x 32 columns]
```

[2]:

```python
# Step 1:
# Dropping irrelevant columns like id.
# Encoding categorical labels (diagnosis) into numeric form.
# Standardizing the data for dimensionality reduction.

# Drop 'id' column
dataset = dataset.drop(columns=['id'])
```

```python
# Encode 'diagnosis' column (M -> 1, B -> 0)
dataset['diagnosis'] = dataset['diagnosis'].map({'M': 1, 'B': 0})

# Separate features and target
X = dataset.drop(columns=['diagnosis'])
y = dataset['diagnosis']

# Standardize the features
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_standardized = scaler.fit_transform(X)
```

[3]:

```python
# Step 2: Apply Dimensionality Reduction (PCA & LDA)
# PCA:
# Use PCA to reduce dimensions while retaining a high percentage of variance (e.g., 95%).

from sklearn.decomposition import PCA

# Perform PCA
pca = PCA(n_components=0.95)
X_pca = pca.fit_transform(X_standardized)

# Check the number of components retained
num_components_pca = pca.n_components_
print(f"Number of components retained by PCA: {num_components_pca}")





from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA

# Perform LDA
lda = LDA(n_components=1)  # Since it's a binary classification problem
X_lda = lda.fit_transform(X_standardized, y)
print(f"LDA reduced the data to {X_lda.shape[1]} component(s).")

# LDA is a supervised dimensionality reduction technique, and the maximum number of components (k) is determined by the number of classes in the dataset.
# The formula is:
#  max =   - 1
# Where C is the number of unique classes.
# Since we are working with a binary classification problem (classes: Malignant and Benign, C=2), the maximum k is 1.
```

```
# This is why we reduced the data to 1 component in LDA.
# LDA ensures that this single component contains the most discriminative␣
↪information for separating the two classes.
```

```
Number of components retained by PCA: 10
LDA reduced the data to 1 component(s).
```

[4]:
```
# Step 3: Compare Models with and Without Dimensionality Reduction
# Train and evaluate logistic regression models:

# Without dimensionality reduction.
# After PCA.
# After LDA.

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix,␣
↪classification_report

# Split the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X_standardized, y,␣
↪test_size=0.2, random_state=42)
X_train_pca, X_test_pca = train_test_split(X_pca, test_size=0.2,␣
↪random_state=42)
X_train_lda, X_test_lda = train_test_split(X_lda, test_size=0.2,␣
↪random_state=42)

# Initialize Logistic Regression
logistic_model = LogisticRegression(max_iter=10000)

# Baseline Model (No Dimensionality Reduction)
logistic_model.fit(X_train, y_train)
y_pred_baseline = logistic_model.predict(X_test)
accuracy_baseline = accuracy_score(y_test, y_pred_baseline)
print(f"Baseline Model Accuracy: {accuracy_baseline * 100:.2f}%")

# PCA Model
logistic_model.fit(X_train_pca, y_train)
y_pred_pca = logistic_model.predict(X_test_pca)
accuracy_pca = accuracy_score(y_test, y_pred_pca)
print(f"PCA Model Accuracy: {accuracy_pca * 100:.2f}%")

# LDA Model
logistic_model.fit(X_train_lda, y_train)
y_pred_lda = logistic_model.predict(X_test_lda)
accuracy_lda = accuracy_score(y_test, y_pred_lda)
print(f"LDA Model Accuracy: {accuracy_lda * 100:.2f}%")
```

```
Baseline Model Accuracy: 97.37%
PCA Model Accuracy: 98.25%
LDA Model Accuracy: 97.37%
```

[5]:
```python
# Step 4: Evaluate Model Performance
# Evaluate performance for each model using confusion matrix, precision,␣
↪recall, and F1-score.

# Function to evaluate model
def evaluate_model(y_true, y_pred, model_name):
    print(f"\n{model_name} Model Evaluation:")
    print("Confusion Matrix:")
    print(confusion_matrix(y_true, y_pred))
    print("\nClassification Report:")
    print(classification_report(y_true, y_pred))

# Evaluate each model
evaluate_model(y_test, y_pred_baseline, "Baseline")
evaluate_model(y_test, y_pred_pca, "PCA")
evaluate_model(y_test, y_pred_lda, "LDA")
```

```
Baseline Model Evaluation:
Confusion Matrix:
[[70  1]
 [ 2 41]]

Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.99      0.98        71
           1       0.98      0.95      0.96        43

    accuracy                           0.97       114
   macro avg       0.97      0.97      0.97       114
weighted avg       0.97      0.97      0.97       114


PCA Model Evaluation:
Confusion Matrix:
[[70  1]
 [ 1 42]]

Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99        71
           1       0.98      0.98      0.98        43
```

```
    accuracy                           0.98       114
   macro avg       0.98      0.98      0.98       114
weighted avg       0.98      0.98      0.98       114


LDA Model Evaluation:
Confusion Matrix:
[[70  1]
 [ 2 41]]

Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.99      0.98        71
           1       0.98      0.95      0.96        43

    accuracy                           0.97       114
   macro avg       0.97      0.97      0.97       114
weighted avg       0.97      0.97      0.97       114
```

[ ]:

```

```

Summary and Conclusion Based on the results:

1. Baseline Model Accuracy: 97.37Confusion Matrix: True Negatives: 70 False Positives: 1 False Negatives: 2 True Positives: 41 Precision, Recall, and F1-Score: Class 0 (No Cancer): Precision: 97Class 1 (Cancer): Precision: 98Observations: High accuracy but uses all 30 features, making the model complex and less interpretable.

2. PCA Model Number of Components Retained: 10 Accuracy: 98.25Confusion Matrix: True Negatives: 70 False Positives: 1 False Negatives: 1 True Positives: 42 Precision, Recall, and F1-Score: Class 0 (No Cancer): Precision: 99Class 1 (Cancer): Precision: 98Observations: PCA achieved the highest accuracy with only 10 components, significantly reducing dimensionality while improving accuracy. Superior precision and recall for both classes compared to the baseline.

3. LDA Model Number of Components Retained: 1 Accuracy: 97.37Confusion Matrix: True Negatives: 70 False Positives: 1 False Negatives: 2 True Positives: 41 Precision, Recall, and F1-Score: Class 0 (No Cancer): Precision: 97Class 1 (Cancer): Precision: 98Observations: LDA simplifies the model to a single component, achieving the same accuracy as the baseline. Excellent interpretability and reduced complexity make it ideal for communicating results to doctors.

Final Recommendation PCA: The best performing model with an accuracy of 98.25LDA: While its accuracy matches the baseline, the simplicity of reducing to just 1 component and its focus on class separability make it a highly interpretable choice for classification tasks. Baseline: High accuracy, but the complexity of using all 30 features makes it less practical for deployment or explanation.

Conclusion: Use PCA for the best accuracy with reduced dimensions. Consider LDA for situations where simplicity and interpretability are the primary goals.PCA and LDA: Designed primarily for dimensionality reduction. PCA is versatile (works for both regression and classification). LDA is tailored for classification, especially when interpretability matters.

DT and RF: Naturally handle feature selection, so dimensionality reduction is often not required. Work well on raw datasets, even with irrelevant features.

Practical Usage: Use PCA or LDA for reducing dimensions before applying regression models or simpler classification algorithms like logistic regression. Use DT or RF directly on raw data unless dimensionality is extremely high, in which case PCA or LDA can still aid preprocessing.

[ ]:
```

```

[ ]:
```

```