PCA_Breast_Cancer

May 29, 2026

1 PRINCIPAL COMPONENT ANALYSIS - BREAST CANCER DATASET

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.pipeline import make_pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix,␣
↪classification_report
```

[4]:
```
data = load_breast_cancer()
X = pd.DataFrame(data.data, columns=data.feature_names)
y = pd.Series(data.target, name="target")

print("Shape:", X.shape)
print("Target classes:", np.unique(y))
print(X.head())
```

```
Shape: (569, 30)
Target classes: [0 1]
   mean radius  mean texture  mean perimeter  mean area  mean smoothness  \
0        17.99         10.38          122.80     1001.0          0.11840
1        20.57         17.77          132.90     1326.0          0.08474
2        19.69         21.25          130.00     1203.0          0.10960
3        11.42         20.38           77.58      386.1          0.14250
4        20.29         14.34          135.10     1297.0          0.10030

   mean compactness  mean concavity  mean concave points  mean symmetry  \
0           0.27760          0.3001              0.14710         0.2419
1           0.07864          0.0869              0.07017         0.1812
2           0.15990          0.1974              0.12790         0.2069
3           0.28390          0.2414              0.10520         0.2597
```

```
4           0.13280          0.1980               0.10430         0.1809

   mean fractal dimension  …  worst radius  worst texture  worst perimeter  \
0                 0.07871  …         25.38          17.33           184.60
1                 0.05667  …         24.99          23.41           158.80
2                 0.05999  …         23.57          25.53           152.50
3                 0.09744  …         14.91          26.50            98.87
4                 0.05883  …         22.54          16.67           152.20

   worst area  worst smoothness  worst compactness  worst concavity  \
0      2019.0            0.1622             0.6656           0.7119
1      1956.0            0.1238             0.1866           0.2416
2      1709.0            0.1444             0.4245           0.4504
3       567.7            0.2098             0.8663           0.6869
4      1575.0            0.1374             0.2050           0.4000

   worst concave points  worst symmetry  worst fractal dimension
0                0.2654          0.4601                  0.11890
1                0.1860          0.2750                  0.08902
2                0.2430          0.3613                  0.08758
3                0.2575          0.6638                  0.17300
4                0.1625          0.2364                  0.07678

[5 rows x 30 columns]
```

[5]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42, stratify=y)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

[6]:
```
pca_full = PCA()
X_train_pca_full = pca_full.fit_transform(X_train_scaled)

explained = pca_full.explained_variance_ratio_
cum_explained = np.cumsum(explained)

print("First 5 explained variance ratios:", np.round(explained[:5], 4))
print("Cumulative variance by first 5:", round(cum_explained[4], 4))
print("Components needed for 95% variance:", np.argmax(cum_explained >= 0.95) +␣
 ↪1)
```

```
First 5 explained variance ratios: [0.4441 0.1894 0.0954 0.0672 0.0552]
Cumulative variance by first 5: 0.8514
Components needed for 95% variance: 10
```

[7]:
```
pca_model = make_pipeline(
    StandardScaler(),
    PCA(n_components=0.95),
    LogisticRegression(max_iter=5000)
)

pca_model.fit(X_train, y_train)
y_pred = pca_model.predict(X_test)

print("Accuracy:", round(accuracy_score(y_test, y_pred), 4))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred,␣
 ↪target_names=data.target_names))

pca_step = pca_model.named_steps["pca"]
print("PCA components retained:", pca_step.n_components_)
```

```
Accuracy: 0.9737
Confusion Matrix:
 [[41  1]
 [ 2 70]]
Classification Report:
              precision    recall  f1-score   support

   malignant       0.95      0.98      0.96        42
      benign       0.99      0.97      0.98        72

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

PCA components retained: 10
```

[8]:
```
baseline_model = make_pipeline(
    StandardScaler(),
    LogisticRegression(max_iter=5000)
)

baseline_cv = cross_val_score(baseline_model, X, y, cv=5, scoring="accuracy").
 ↪mean()
pca_cv = cross_val_score(pca_model, X, y, cv=5, scoring="accuracy").mean()

print(f"Baseline (no PCA) mean CV accuracy: {baseline_cv:.4f}")
print(f"With PCA (95% variance) mean CV accuracy: {pca_cv:.4f}")
```

```
Baseline (no PCA) mean CV accuracy: 0.9807
With PCA (95% variance) mean CV accuracy: 0.9807
```

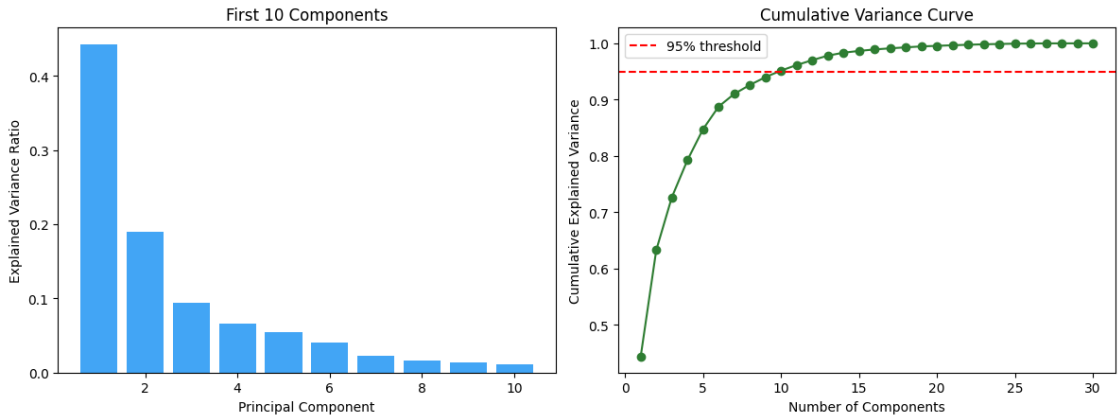
[9]:
```
pca_2 = PCA(n_components=2)
X_scaled = StandardScaler().fit_transform(X)
pca_2.fit(X_scaled)

loading_df = pd.DataFrame(
    pca_2.components_.T,
    columns=["PC1_loading", "PC2_loading"],
    index=X.columns
)

top_pc1 = loading_df["PC1_loading"].abs().sort_values(ascending=False).head(8)
print("Top contributors to PC1:")
print(loading_df.loc[top_pc1.index].sort_values("PC1_loading", key=np.abs,␣
 ↪ascending=False))
```

```
Top contributors to PC1:
                      PC1_loading  PC2_loading
mean concave points      0.260854    -0.034768
mean concavity           0.258400     0.060165
worst concave points     0.250886    -0.008257
mean compactness         0.239285     0.151892
worst perimeter          0.236640    -0.199878
worst concavity          0.228768     0.097964
worst radius             0.227997    -0.219866
mean perimeter           0.227537    -0.215181
```

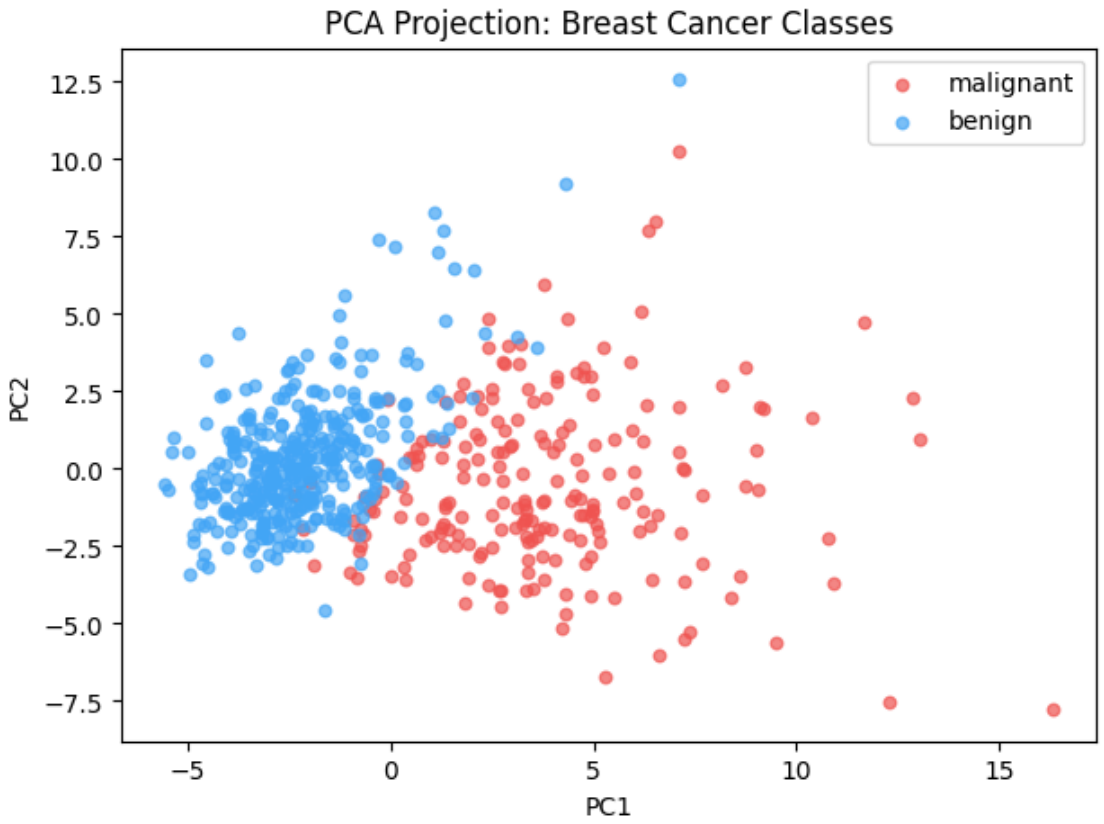
2 GRAPHS

[10]:
```
pca_vis = PCA()
X_scaled = StandardScaler().fit_transform(X)
pca_vis.fit(X_scaled)

exp = pca_vis.explained_variance_ratio_
cum = np.cumsum(exp)

fig, axes = plt.subplots(1, 2, figsize=(12, 4.6))

axes[0].bar(range(1, 11), exp[:10], color="#42a5f5")
axes[0].set_xlabel("Principal Component")
axes[0].set_ylabel("Explained Variance Ratio")
axes[0].set_title("First 10 Components")

axes[1].plot(range(1, len(cum) + 1), cum, marker="o", linewidth=1.5,␣
 ↪color="#2e7d32")
axes[1].axhline(0.95, color="red", linestyle="--", label="95% threshold")
axes[1].set_xlabel("Number of Components")
axes[1].set_ylabel("Cumulative Explained Variance")
```

```
axes[1].set_title("Cumulative Variance Curve")
axes[1].legend()

plt.tight_layout()
plt.show()
```



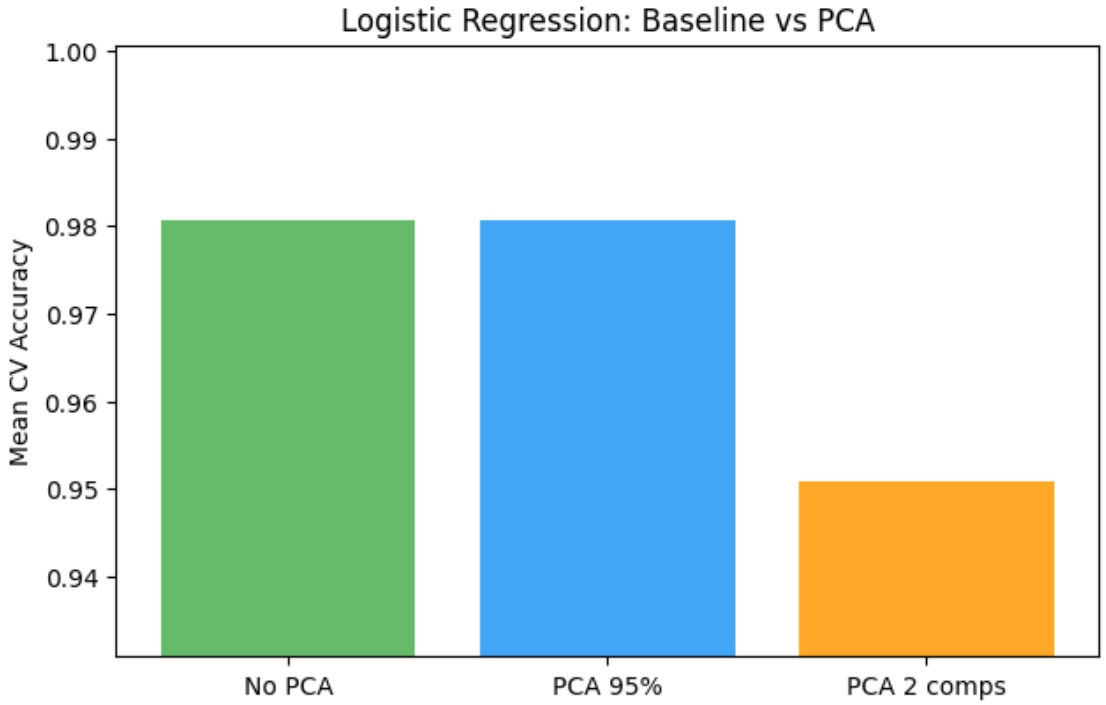
[11]:
```
X_scaled = StandardScaler().fit_transform(X)
X_2d = PCA(n_components=2).fit_transform(X_scaled)

plt.figure(figsize=(7, 5))
plt.scatter(X_2d[y == 0, 0], X_2d[y == 0, 1], s=22, alpha=0.7, label=data.
 ↪target_names[0], color="#ef5350")
plt.scatter(X_2d[y == 1, 0], X_2d[y == 1, 1], s=22, alpha=0.7, label=data.
 ↪target_names[1], color="#42a5f5")
plt.xlabel("PC1")
plt.ylabel("PC2")
plt.title("PCA Projection: Breast Cancer Classes")
plt.legend()
plt.show()
```



[12]:
```
models = {
    "No PCA": make_pipeline(StandardScaler(),␣
 ↪LogisticRegression(max_iter=5000)),
    "PCA 95%": make_pipeline(StandardScaler(), PCA(n_components=0.95),␣
 ↪LogisticRegression(max_iter=5000)),
    "PCA 2 comps": make_pipeline(StandardScaler(), PCA(n_components=2),␣
 ↪LogisticRegression(max_iter=5000))
}

rows = []
for name, model in models.items():
    score = cross_val_score(model, X, y, cv=5, scoring="accuracy").mean()
    rows.append((name, score))

comp = pd.DataFrame(rows, columns=["model", "cv_accuracy"]).
 ↪sort_values("cv_accuracy", ascending=False)
print(comp)

plt.figure(figsize=(7.2, 4.5))
```

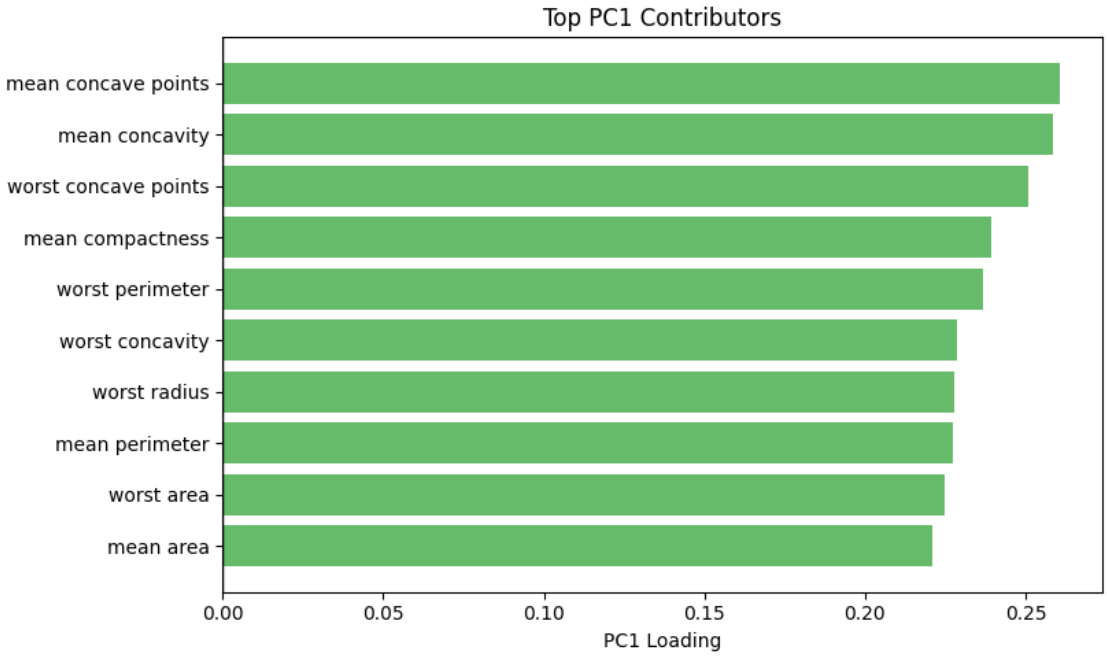
```
plt.bar(comp["model"], comp["cv_accuracy"], color=["#66bb6a", "#42a5f5",␣
  ↪"#ffa726"])
plt.ylabel("Mean CV Accuracy")
plt.title("Logistic Regression: Baseline vs PCA")
plt.ylim(comp["cv_accuracy"].min() - 0.02, comp["cv_accuracy"].max() + 0.02)
plt.show()
```

```
          model  cv_accuracy
0       No PCA     0.980686
1      PCA 95%     0.980671
2  PCA 2 comps     0.950846
```



[13]:
```
pca_2 = PCA(n_components=2)
X_scaled = StandardScaler().fit_transform(X)
pca_2.fit(X_scaled)

load = pd.DataFrame({
    "feature": X.columns,
    "pc1_loading": pca_2.components_[0]
})
load["abs_loading"] = load["pc1_loading"].abs()
top = load.sort_values("abs_loading", ascending=False).head(10).
  ↪sort_values("pc1_loading")
```

```
colors = ["#ef5350" if v < 0 else "#66bb6a" for v in top["pc1_loading"]]

plt.figure(figsize=(8.3, 5))
plt.barh(top["feature"], top["pc1_loading"], color=colors)
plt.axvline(0, color="black", linewidth=1)
plt.xlabel("PC1 Loading")
plt.title("Top PC1 Contributors")
plt.tight_layout()
plt.show()
```



[ ]: