Q1. PCA for Breast Cancer Diagnosis (Real Data) (4×1 = 4 marks)

🧠 Context

The Wisconsin Breast Cancer dataset contains 30 features per biopsy.

🎯 Goal

Use Principal Component Analysis (PCA) to denoise the data and visualize class separability.

📂 Data Source

Use either of the following:

- `sklearn.datasets.load_breast_cancer()`
- Kaggle copy: Breast Cancer Wisconsin Dataset

In [85]:
```
import numpy as np
import pandas as pd
from sklearn import datasets
data = datasets.load_breast_cancer()
```

In [86]:
```
data.keys()
```

Out[86]:
```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'f
ilename', 'data_module'])
```

In [87]:
```
data.data.shape, data.target.shape
```

Out[87]:
```
((569, 30), (569,))
```

In [88]:
```
df = pd.DataFrame(data.data, columns=data.feature_names)
df['target']=data.target
print(df.shape)
print(df.head())
```

```
(569, 31)
   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 texture  worst perimeter  worst area  \
0                 0.07871  ...          17.33           184.60      2019.0
1                 0.05667  ...          23.41           158.80      1956.0
2                 0.05999  ...          25.53           152.50      1709.0
3                 0.09744  ...          26.50            98.87       567.7
4                 0.05883  ...          16.67           152.20      1575.0

   worst smoothness  worst compactness  worst concavity  worst concave points  \
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

   worst symmetry  worst fractal dimension  target
0          0.4601                  0.11890       0
1          0.2750                  0.08902       0
2          0.3613                  0.08758       0
3          0.6638                  0.17300       0
4          0.2364                  0.07678       0

[5 rows x 31 columns]
```

🔍 Tasks

a) Standardize & Covariance

- Standardize all 30 features to have zero mean and unit variance.
- Compute the 30 × 30 sample covariance matrix (S).
- Report the Frobenius norm (|S|_F) as a numeric value.

In [89]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_std = scaler.fit_transform(data.data)
df2 = pd.DataFrame(X_std, columns=data.feature_names)
pd.concat([df2.mean(),df2.std()],axis=1, keys=['mean','std'])
```

Out[89]:

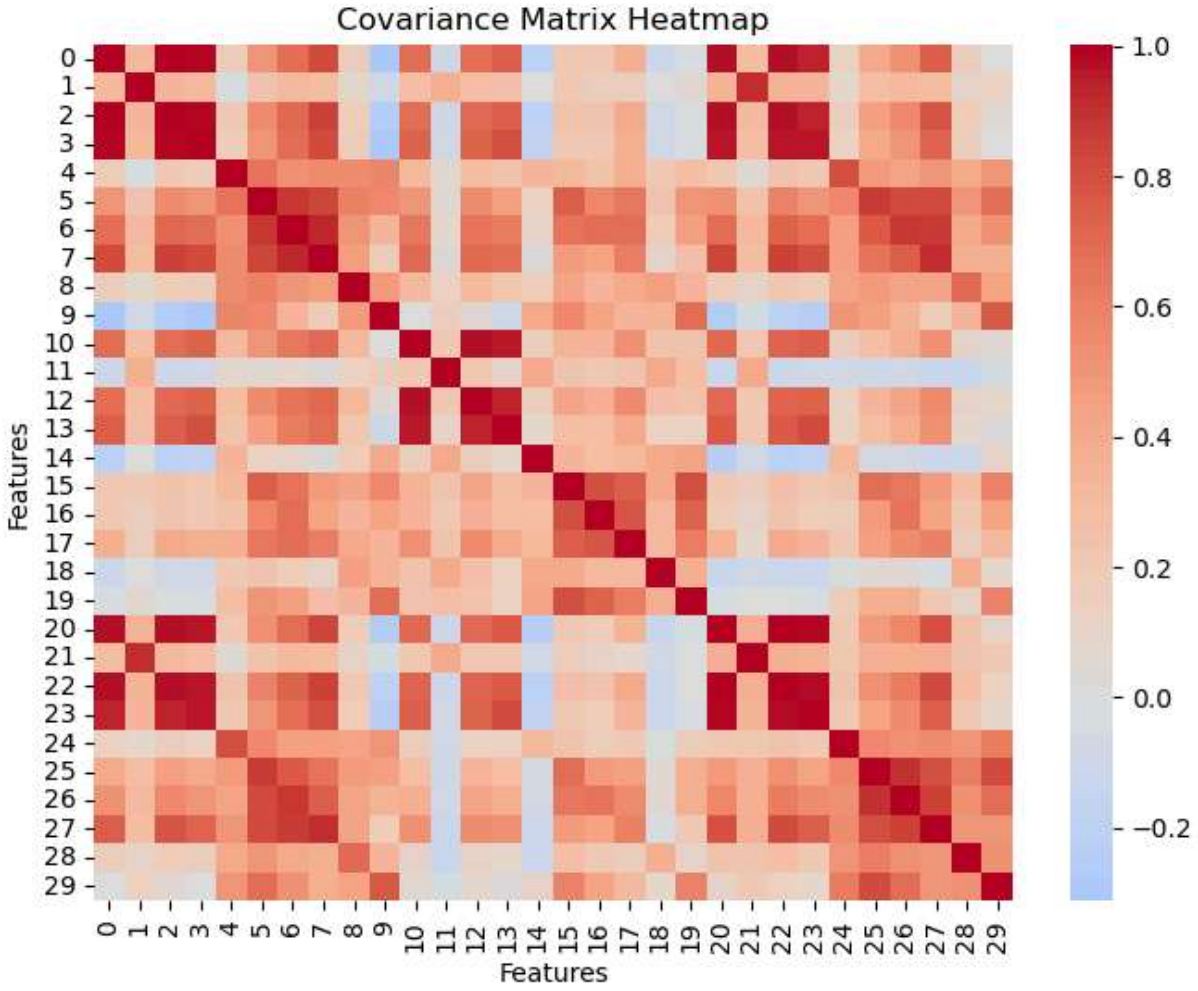
	mean	std
mean radius	-3.162867e-15	1.00088
mean texture	-6.530609e-15	1.00088
mean perimeter	-7.078891e-16	1.00088
mean area	-8.799835e-16	1.00088
mean smoothness	6.132177e-15	1.00088
mean compactness	-1.120369e-15	1.00088
mean concavity	-4.421380e-16	1.00088
mean concave points	9.732500e-16	1.00088
mean symmetry	-1.971670e-15	1.00088
mean fractal dimension	-1.453631e-15	1.00088
radius error	-9.076415e-16	1.00088
texture error	-8.853492e-16	1.00088
perimeter error	1.773674e-15	1.00088
area error	-8.291551e-16	1.00088
smoothness error	-7.541809e-16	1.00088
compactness error	-3.921877e-16	1.00088
concavity error	7.917900e-16	1.00088
concave points error	-2.739461e-16	1.00088
symmetry error	-3.108234e-16	1.00088
fractal dimension error	-3.366766e-16	1.00088
worst radius	-2.333224e-15	1.00088
worst texture	1.763674e-15	1.00088
worst perimeter	-1.198026e-15	1.00088
worst area	5.049661e-16	1.00088
worst smoothness	-5.213170e-15	1.00088
worst compactness	-2.174788e-15	1.00088
worst concavity	6.856456e-16	1.00088
worst concave points	-1.412656e-16	1.00088
worst symmetry	-2.289567e-15	1.00088
worst fractal dimension	2.575171e-15	1.00088

In [90]:
```
df2.mean().mean(), df2.std().mean()
```

Out[90]:
```
(np.float64(-6.369603796435407e-16), np.float64(1.000879894582902))
```

In [91]:
```
# covariance matrix
S = np.cov(df2, rowvar=False)
import matplotlib.pyplot as plt
import seaborn as sns

# printing heat map of covariance matrix
plt.figure(figsize=(8, 6))
sns.heatmap(S, annot=False, cmap='coolwarm', center=0)
plt.title("Covariance Matrix Heatmap")
plt.xlabel("Features")
plt.ylabel("Features")
plt.show()
```



In [92]:
```
# Frobenius norm
fro_norm = np.linalg.norm(S, 'fro')
print("Frobenius norm of covariance matrix:", fro_norm)
```

```
Frobenius norm of covariance matrix: 15.062350986709806
```

b) Top-k Eigenpairs

- Set (k = 2).
- Use `sklearn.decomposition.PCA` to compute:
  - (i) Top two eigenvalues (\lambda_1, \lambda_2)
  - (ii) Explained variance ratio (%) of each
  - (iii) Cumulative explained variance (%)

In [93]:
```
# Eigvne value calculation as a reference, eigen values are printed in descending o
eigenvalues, eigenvectors = np.linalg.eig(S)
np.set_printoptions(precision=4, suppress=True)
idx = np.argsort(eigenvalues)[::-1]
print(eigenvalues[idx])
```

```
[13.305   5.7014  2.8229  1.9841  1.6516  1.2095  0.6764  0.4775  0.4176
 0.3513  0.2944  0.2616  0.2418  0.1573  0.0943  0.08    0.0595  0.0527
 0.0496  0.0312  0.03    0.0275  0.0244  0.0181  0.0155  0.0082  0.0069
 0.0016  0.0008  0.0001]
```

In [94]:
```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
#df2 is the normalized breast cancer data (mean=0, sd = 1)
pca.fit(df2)
eigenvalues = pca.explained_variance_
print("Top 2 eigen values: ", eigenvalues)
explained_variance_ratio = pca.explained_variance_ratio_ * 100
print("Explained variance ratio: ", explained_variance_ratio)
cumulative_variance = explained_variance_ratio.cumsum()
print("Cumulative variance ratio: ", cumulative_variance)
```

```
Top 2 eigen values:  [13.305   5.7014]
Explained variance ratio:  [44.272  18.9712]
Cumulative variance ratio:  [44.272  63.2432]
```

First 2 eigen values are covering up to ~82% of variance in the data

In [95]:
```
help(pca.fit)
```

```
Help on method fit in module sklearn.decomposition._pca:

fit(X, y=None) method of sklearn.decomposition._pca.PCA instance
    Fit the model with X.

    Parameters
    ----------
    X : {array-like, sparse matrix} of shape (n_samples, n_features)
        Training data, where `n_samples` is the number of samples
        and `n_features` is the number of features.

    y : Ignored
        Ignored.

    Returns
    -------
    self : object
        Returns the instance itself.
```

c) 2D Projection

- Project all samples onto the top-2 principal components.
- Plot a labeled 2D scatter plot (malignant vs. benign).
- Report the class means in this 2D space.

In [96]:
```
# projecting samples onto top-2 principal components
X_pca = pca.fit_transform(df2)
df_pca = pd.DataFrame(X_pca, columns=['PC1','PC2'])
df_pca['target']= data.target
print(df_pca.head())
```

```
        PC1        PC2  target
0  9.192837   1.948583       0
1  2.387802  -3.768172       0
2  5.733896  -1.075174       0
3  7.122953  10.275589       0
4  3.935302  -1.948072       0
```

In [97]:
```
X_pca.shape
```

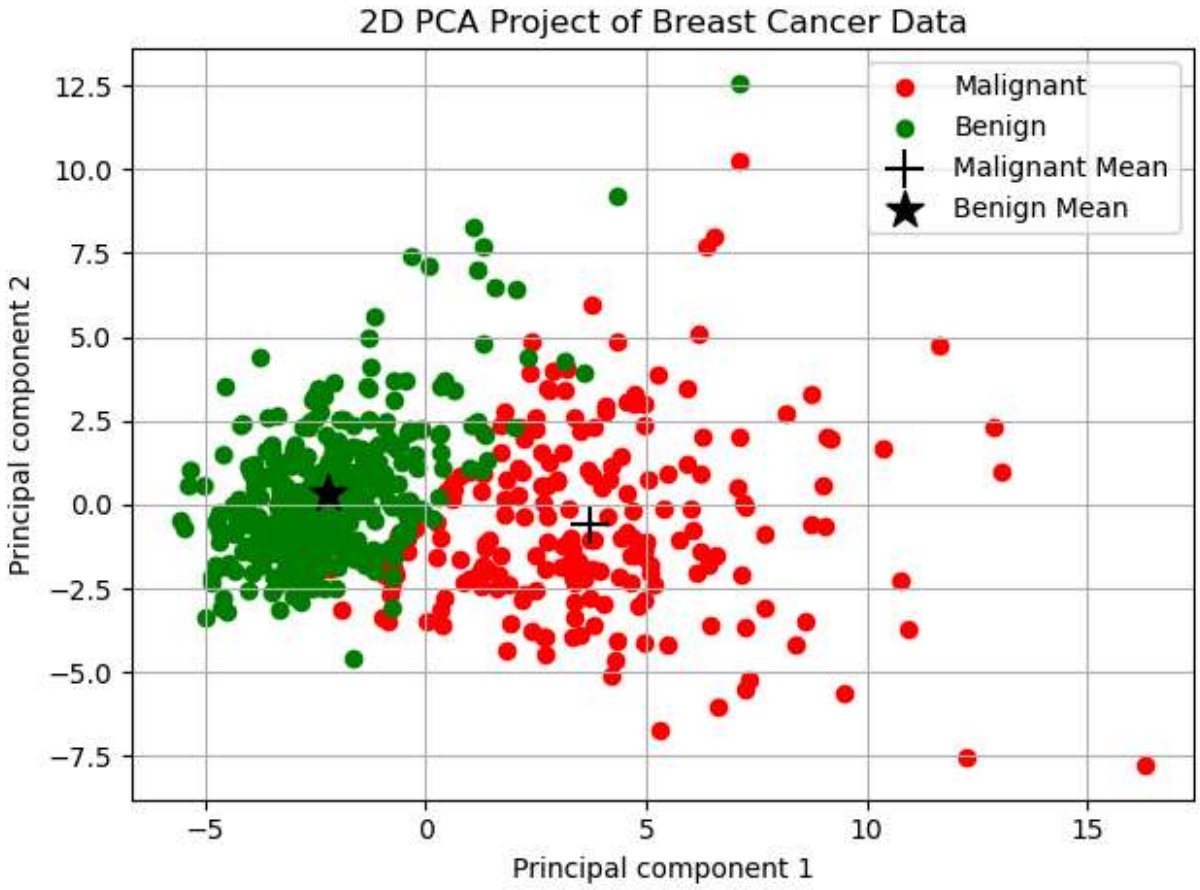
Out[97]:
```
(569, 2)
```

In [98]:
```
malignant_features = df_pca[df_pca['target']==0][['PC1','PC2']]
benign_features = df_pca[df_pca['target']==1][['PC1','PC2']]
malignant_mean = malignant_features.mean().values
benign_mean = benign_features.mean().values
print("Malignant mean (PC1, PC2):", malignant_mean)
print("Benign mean (PC1, PC2):", benign_mean)

plt.scatter(malignant_features['PC1'], malignant_features['PC2'], c='red', label='M
plt.scatter(benign_features['PC1'], benign_features['PC2'], c='green', label='Benig

plt.scatter(*malignant_mean, c='k', marker='+', s=200, label='Malignant Mean')
plt.scatter(*benign_mean, c='k', marker='*', s=200, label='Benign Mean')
plt.xlabel('Principal component 1')
plt.ylabel('Principal component 2')
plt.title('2D PCA Project of Breast Cancer Data')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

```
Malignant mean (PC1, PC2): [ 3.7148 -0.5831]
Benign mean (PC1, PC2): [-2.206   0.3463]
```



d) Reconstruction Error

- Inverse-transform the 2D projection back to the original 30D space.
- Report the average per-sample reconstruction Mean Squared Error (MSE) as a numeric value.

In [99]:
```
X_recon = pca.inverse_transform(X_pca)
X_recon.shape
```

Out[99]:
```
(569, 30)
```

In [100…
```
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(X_std, X_recon)
print(f"Average per-sample reconstruction MSE: {mse:.4f}")
```

```
Average per-sample reconstruction MSE: 0.3676
```

==========

In [ ]: