Experiment 2: GMM Clustering

Imports and Configuration

```
import os
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import time
%cd ..
from src.models.gmm import GMM, Covariance
from src.utils.metrics import *
from src.utils.utils import *

RANDOM_SEED = 42
np.random.seed(RANDOM_SEED)
```

```
c:\Users\ziade\OneDrive\Desktop\Term 7\ML\Lab 4\Assignment4-
Unsupervised-Clustering-Analysis
```

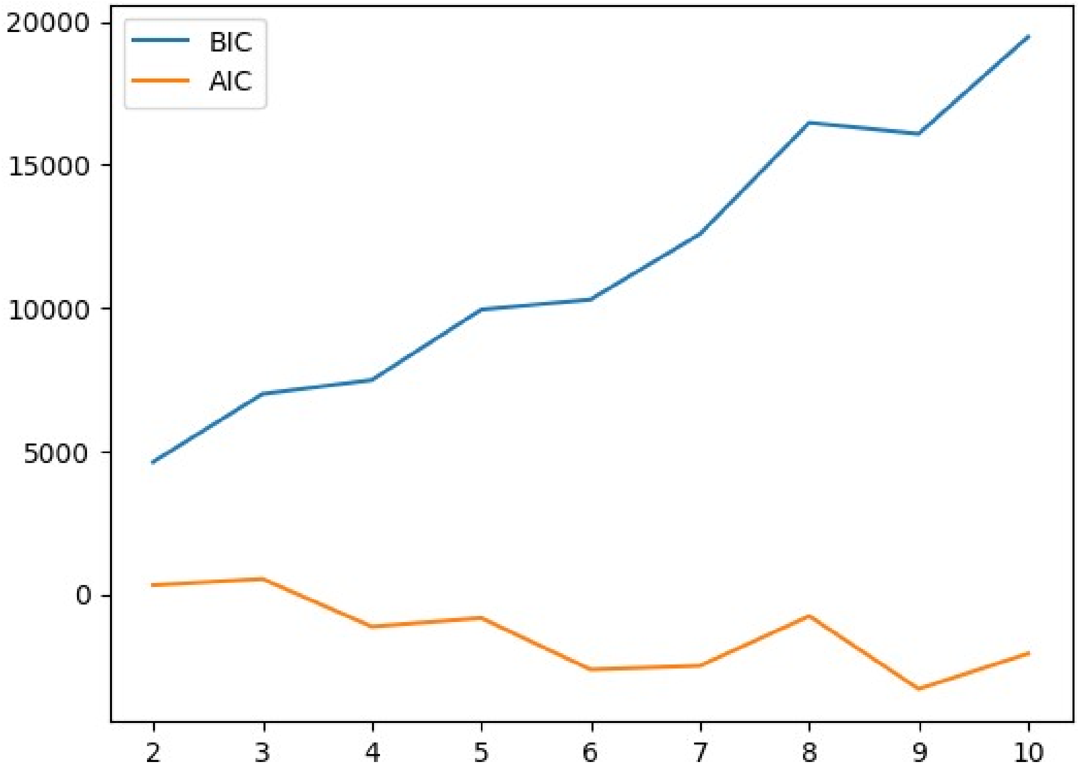
Data Loading and Preprocessing

```
X_scaled, y_true =
load_scale_data(data_path=os.path.abspath('./data/breast_cancer.csv'))

print(f"Dataset shape: {X_scaled.shape}")
print(f"Labels shape: {y_true.shape}")
```

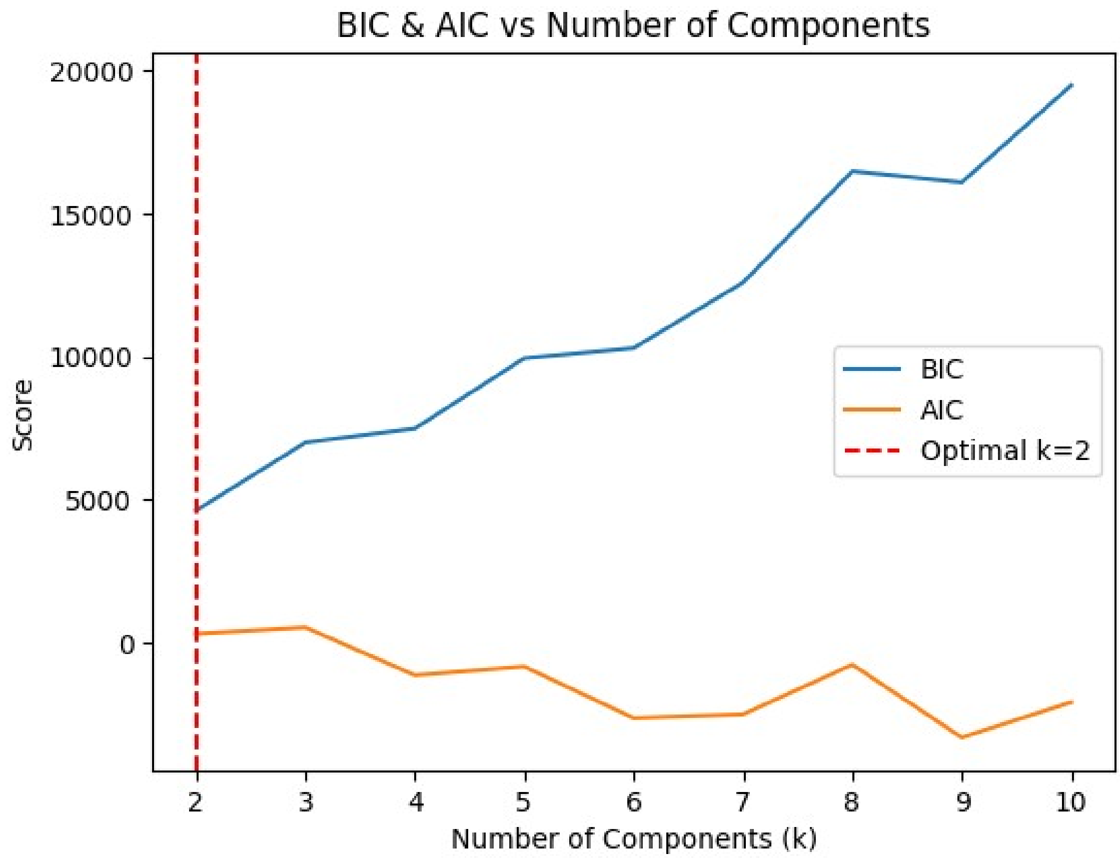
```
Dataset shape: (569, 30)
Labels shape: (569,)
```

Finding Optimal Components using BIC and AIC

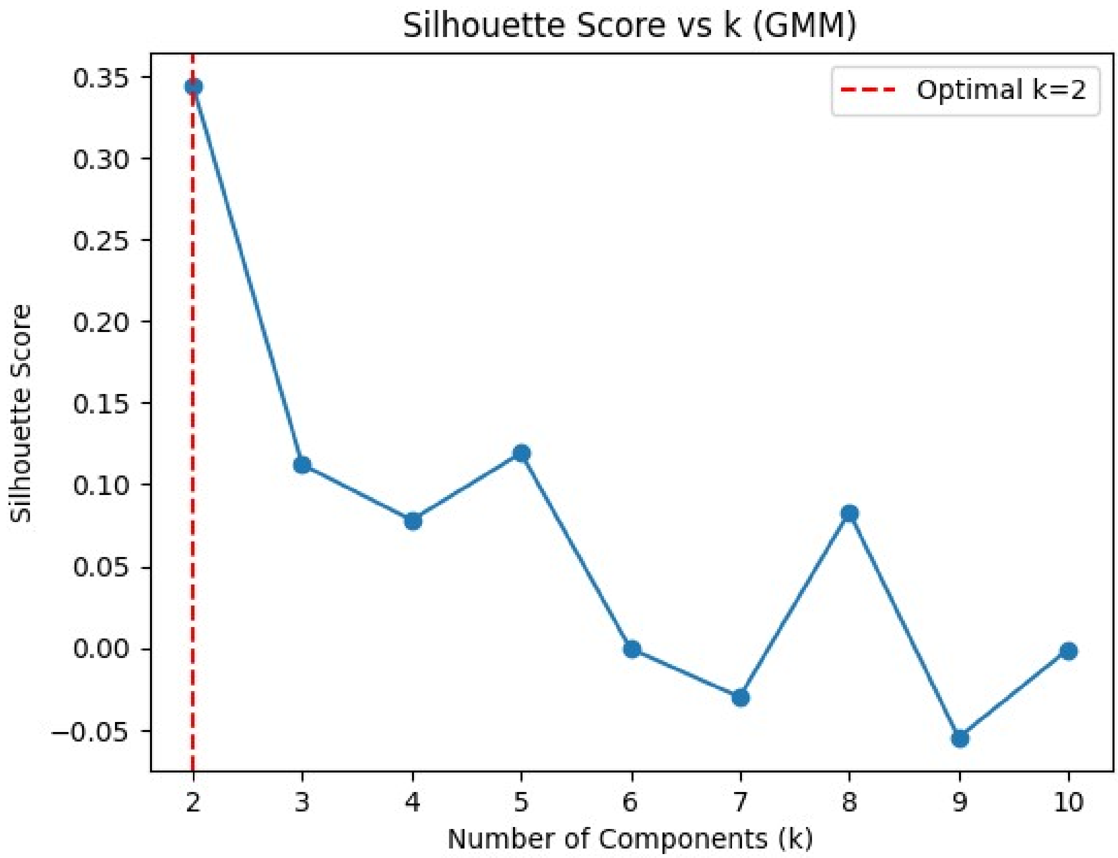
```
k_range = range(2, 11)
bic_scores = []
aic_scores = []

for k in k_range:
    gmm = GMM(k=k, covariance_type=Covariance.FULL)
    gmm.fit(X_scaled)
    bic_scores.append(gmm.bic(X_scaled))
    aic_scores.append(gmm.aic(X_scaled))

plt.plot(k_range, bic_scores, label='BIC')
plt.plot(k_range, aic_scores, label='AIC')
plt.legend()
plt.show()
```




```
optimal_k = k_range[np.argmin(bic_scores)]

plt.figure()
plt.plot(k_range, bic_scores, label='BIC')
plt.plot(k_range, aic_scores, label='AIC')
plt.axvline(optimal_k, color='red', linestyle='--', label=f'Optimal
k={optimal_k}')
plt.xlabel("Number of Components (k)")
plt.ylabel("Score")
plt.title("BIC & AIC vs Number of Components")
plt.legend()
plt.show()
```




```
silhouette_scores = []

for k in k_range:
    gmm = GMM(k=k, covariance_type=Covariance.FULL)
    gmm.fit(X_scaled)
    labels = gmm.predict(X_scaled)
    silhouette_scores.append(compute_silhouette_score(X_scaled,
labels))

plt.figure()
plt.plot(k_range, silhouette_scores, marker='o')
plt.axvline(optimal_k, color='red', linestyle='--', label=f'Optimal
k={optimal_k}')
plt.xlabel("Number of Components (k)")
plt.ylabel("Silhouette Score")
plt.title("Silhouette Score vs k (GMM)")
plt.legend()
plt.show()
```

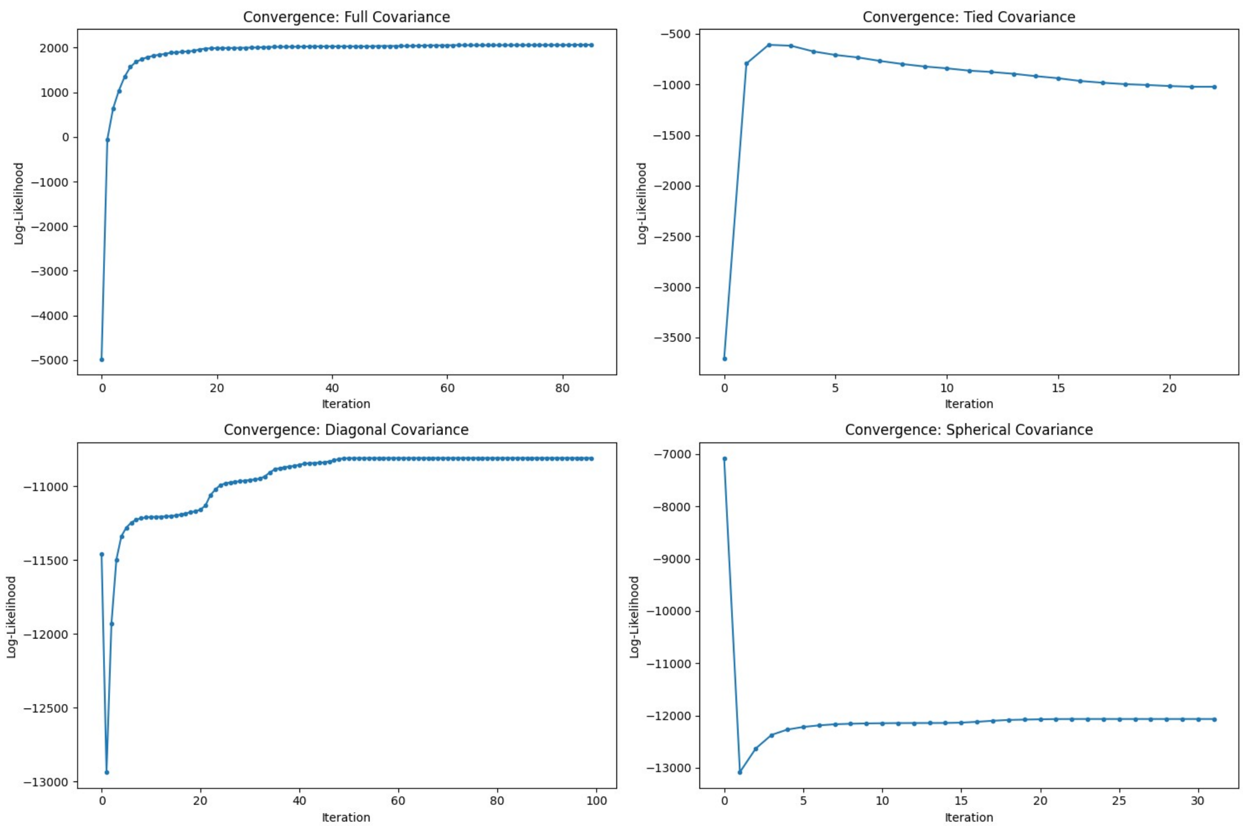


Comparing Covariance Types

`Plots`

```
optimal_k = 2

cov_types = [Covariance.FULL, Covariance.TIED, Covariance.DIAGONAL,
Covariance.SPHERICAL]
results = {}

plt.figure(figsize=(15, 10))

for i, cov_type in enumerate(cov_types):
    start_time = time.time()
    gmm = GMM(k=optimal_k, covariance_type=cov_type, max_iter=100,
tol=1e-4)
    gmm.fit(X_scaled)
    elapsed_time = time.time() - start_time

    results[cov_type.value] = {
        'Final Log-Likelihood': gmm.log_likelihood_[-1],
        'Iterations': len(gmm.log_likelihood_),
        'Time (s)': elapsed_time,
```

```python
        'BIC': gmm.bic(X_scaled),
        'AIC': gmm.aic(X_scaled)
    }

    # Plot Log-Likelihood Convergence
    plt.subplot(2, 2, i+1)
    plt.plot(gmm.log_likelihood_, marker='.')
    plt.title(f"Convergence: {cov_type.value.capitalize()}
Covariance")
    plt.xlabel("Iteration")
    plt.ylabel("Log-Likelihood")

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



Comparison Table

```python
df_results = pd.DataFrame(results).T
print("Covariance Type Comparison:")
print(df_results)
```

```
Covariance Type Comparison:
           Final Log-Likelihood  Iterations  Time (s)          BIC  \
full                2062.197856        86.0  0.075949  2162.389798
```

```
tied                  -1023.247417        23.0  0.013904   5383.375942
diagonal             -10807.157434       100.0  0.043878  22381.924401
spherical            -12064.975390        32.0  0.017867  24529.615248

                    AIC
full       -2142.395713
tied        3098.494833
diagonal   21856.314868
spherical  24255.950781
```

GMM Performance Analysis

Select the Best Model

```python
best_cov = df_results['BIC'].idxmin()
print(f"Analyzing Best Model: {best_cov} covariance")

best_gmm = GMM(k=optimal_k, covariance_type=Covariance(best_cov))
best_gmm.fit(X_scaled)
y_gmm_pred = best_gmm.predict(X_scaled)
```

```
Analyzing Best Model: full covariance
```

Internal Validation Metrics

```python
sil_score = compute_silhouette_score(X_scaled, y_gmm_pred)
print(f"Internal Metrics:")
print(f" - Silhouette Score: {sil_score:.4f}")
print(f" - Final Log-Likelihood: {best_gmm.log_likelihood_[-1]:.4f}")
print(f" - BIC: {best_gmm.bic(X_scaled):.4f}")
```

```
Internal Metrics:
 - Silhouette Score: 0.2351
 - Final Log-Likelihood: -228.1930
 - BIC: 6743.1715
```

External Validation Metrics

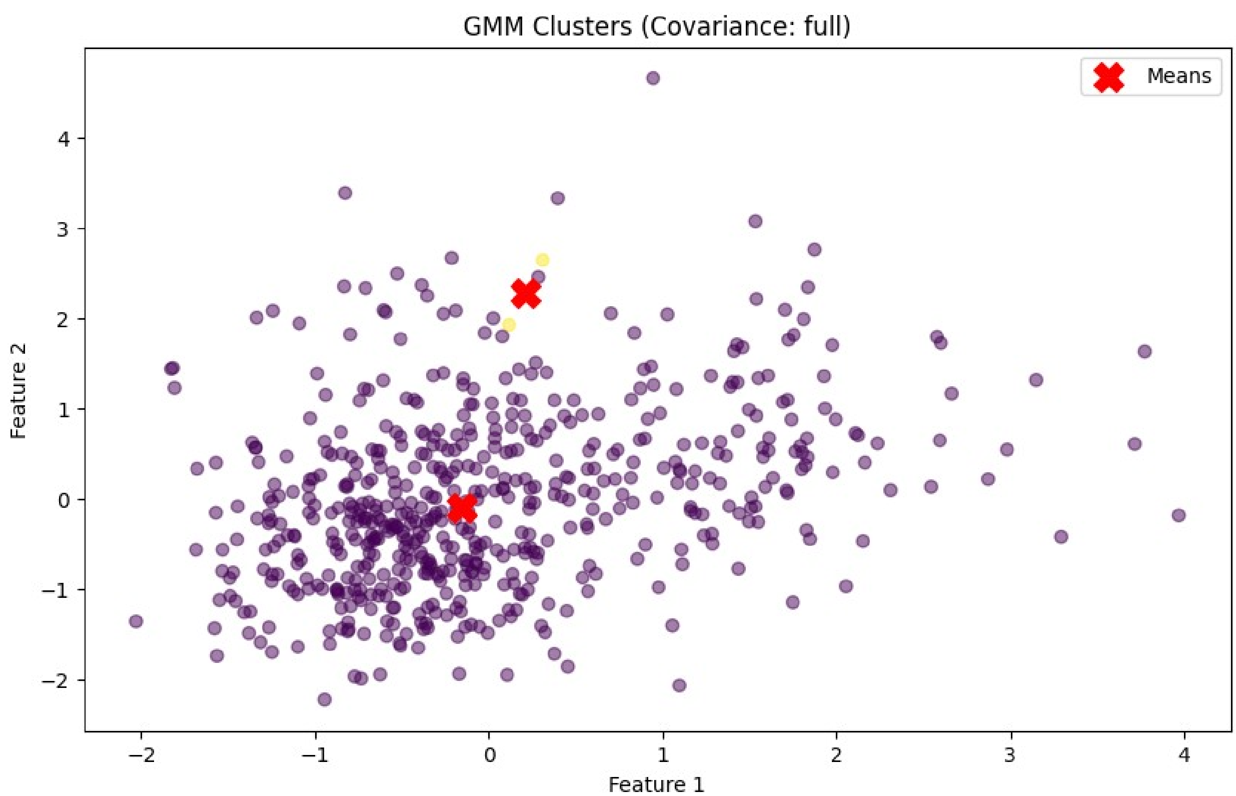
```python
# Map GMM clusters to true labels
y_gmm_aligned = align_clusters_with_labels(y_true, y_gmm_pred)

purity = purity_score(y_true, y_gmm_aligned)

print(f"\nExternal Metrics (Labels used for evaluation only):")
print(f" - Purity: {purity:.4f}")

ari = compute_ari(y_true, y_gmm_aligned)
nmi = compute_nmi(y_true, y_gmm_aligned)
```

```
print(f" - ARI: {ari:.4f}")
print(f" - NMI: {nmi:.4f}")
```

```
External Metrics (Labels used for evaluation only):
 - Purity: 0.6309
 - ARI: 0.0048
 - NMI: 0.0102
```

Cluster assignments on 2D Projections

```
plt.figure(figsize=(10, 6))
# Using first two features for 2D visualization
plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=y_gmm_pred,
cmap='viridis', alpha=0.5)
plt.scatter(best_gmm.mu[:, 0], best_gmm.mu[:, 1], c='red', marker='X',
s=200, label='Means')
plt.title(f"GMM Clusters (Covariance: {best_cov})")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.legend()
plt.show()
```



Confusion Matrix Analysis

```
cm_gmm = compute_confusion_matrix(y_true, y_gmm_aligned)
plt.figure(figsize=(5, 4))
sns.heatmap(cm_gmm, annot=True, fmt='d', cmap='Greens')
plt.title("Confusion Matrix: GMM (Best Model)")
plt.ylabel("True Label")
plt.xlabel("Predicted Label")
plt.show()
```