Code

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler, RobustScaler
from sklearn.decomposition import PCA
from sklearn.mixture import GaussianMixture
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

In [2]:
```
columns = ['pelvic_incidence', 'pelvic_tilt', 'lumbar_lordosis_angle', 'sacral_slope', 'pelvic_radius', 'grade_of_spondylolisthesis', 'class']
data = pd.read_csv('vertebral_column_data.txt', header=None, names=columns, sep=' ')
data.head()
```

Out[2]:

	pelvic_incidence	pelvic_tilt	lumbar_lordosis_angle	sacral_slope	pelvic_radius	grade_of_spondylolisthesis	class
0	63.03	22.55	39.61	40.48	98.67	-0.25	AB
1	39.06	10.06	25.02	29.00	114.41	4.56	AB
2	68.83	22.22	50.09	46.61	105.99	-3.53	AB
3	69.30	24.65	44.31	44.64	101.87	11.21	AB
4	49.71	9.65	28.32	40.06	108.17	7.92	AB

Exploratory Data Analysis for Vertebral Column Data

In [5]:
```
data.describe()
```

Out[5]:

	pelvic_incidence	pelvic_tilt	lumbar_lordosis_angle	sacral_slope	pelvic_radius	grade_of_spondylolisthesis
count	310.000000	310.000000	310.000000	310.000000	310.000000	310.000000
mean	60.496484	17.542903	51.930710	42.953871	117.920548	26.296742
std	17.236109	10.008140	18.553766	13.422748	13.317629	37.558883
min	26.150000	-6.550000	14.000000	13.370000	70.080000	-11.060000
25%	46.432500	10.667500	37.000000	33.347500	110.710000	1.600000
50%	58.690000	16.360000	49.565000	42.405000	118.265000	11.765000
75%	72.880000	22.120000	63.000000	52.692500	125.467500	41.285000
max	129.830000	49.430000	125.740000	121.430000	163.070000	418.540000

In [6]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 310 entries, 0 to 309
Data columns (total 7 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   pelvic_incidence            310 non-null    float64
 1   pelvic_tilt                 310 non-null    float64
 2   lumbar_lordosis_angle       310 non-null    float64
 3   sacral_slope                310 non-null    float64
 4   pelvic_radius               310 non-null    float64
 5   grade_of_spondylolisthesis  310 non-null    float64
 6   class                       310 non-null    object 
dtypes: float64(6), object(1)
memory usage: 17.1+ KB
```

In [7]:
```
data.isnull().sum()
```

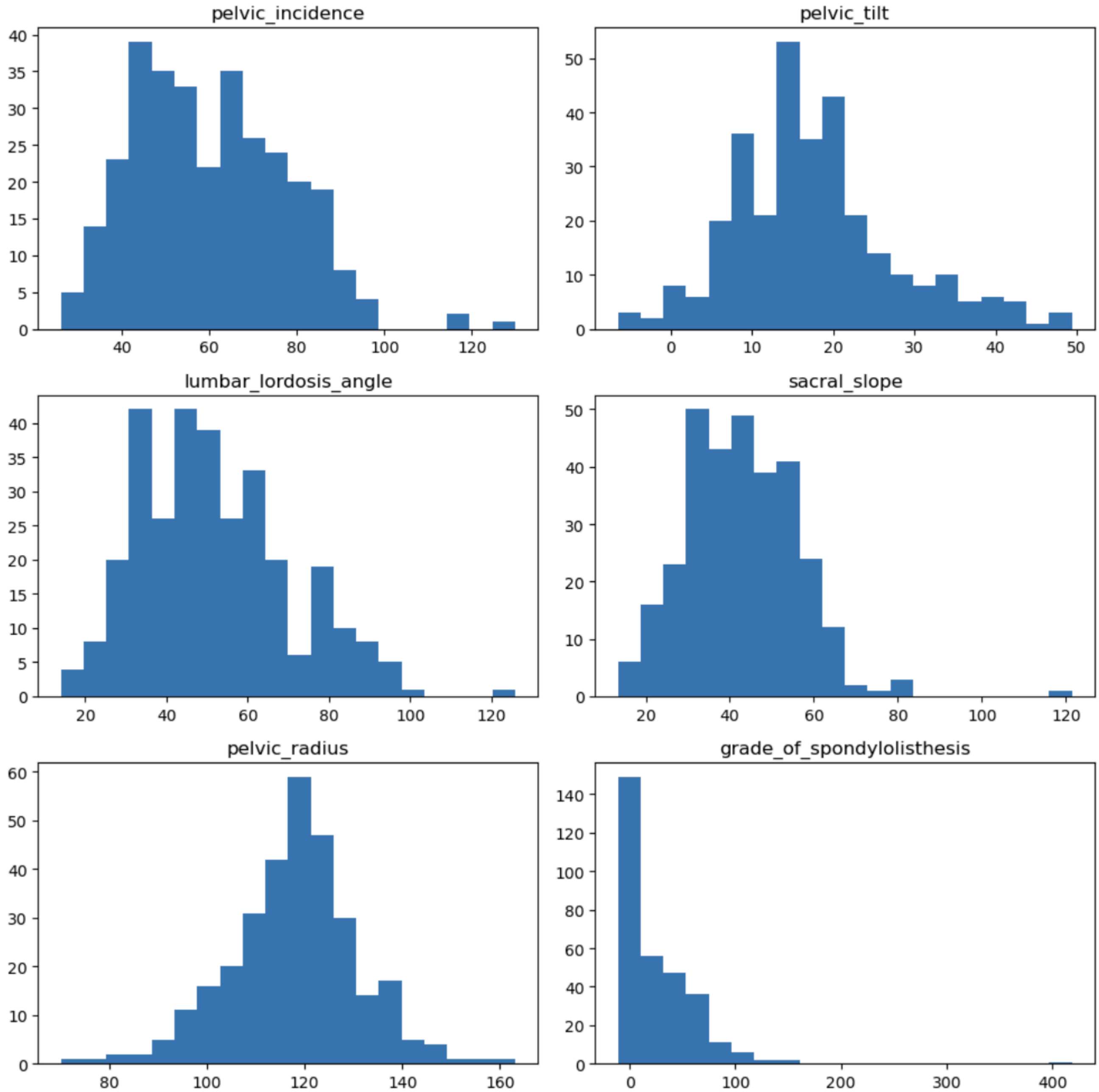
Out[7]:
```
pelvic_incidence              0
pelvic_tilt                   0
lumbar_lordosis_angle         0
sacral_slope                  0
pelvic_radius                 0
grade_of_spondylolisthesis    0
class                         0
dtype: int64
```

In [12]:
```
data.duplicated()
```

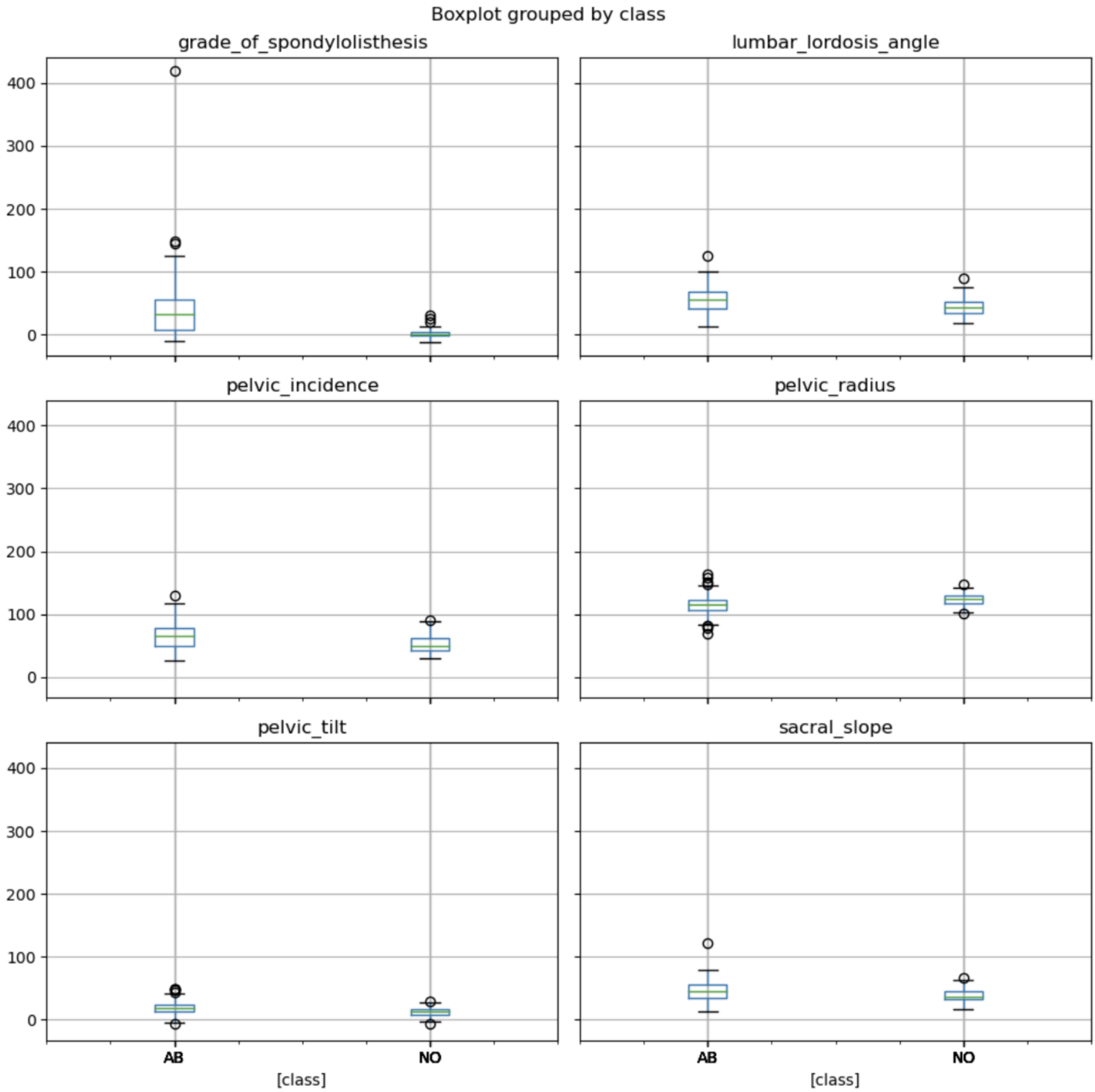
Out[12]:
```
0      False
1      False
2      False
3      False
4      False
       ...  
305    False
306    False
307    False
308    False
309    False
Length: 310, dtype: bool
```

Data visualisation

In [8]:
```
# Histogram

data.hist(bins=20, figsize=(10, 10), grid=False)
plt.tight_layout()
plt.show()
```

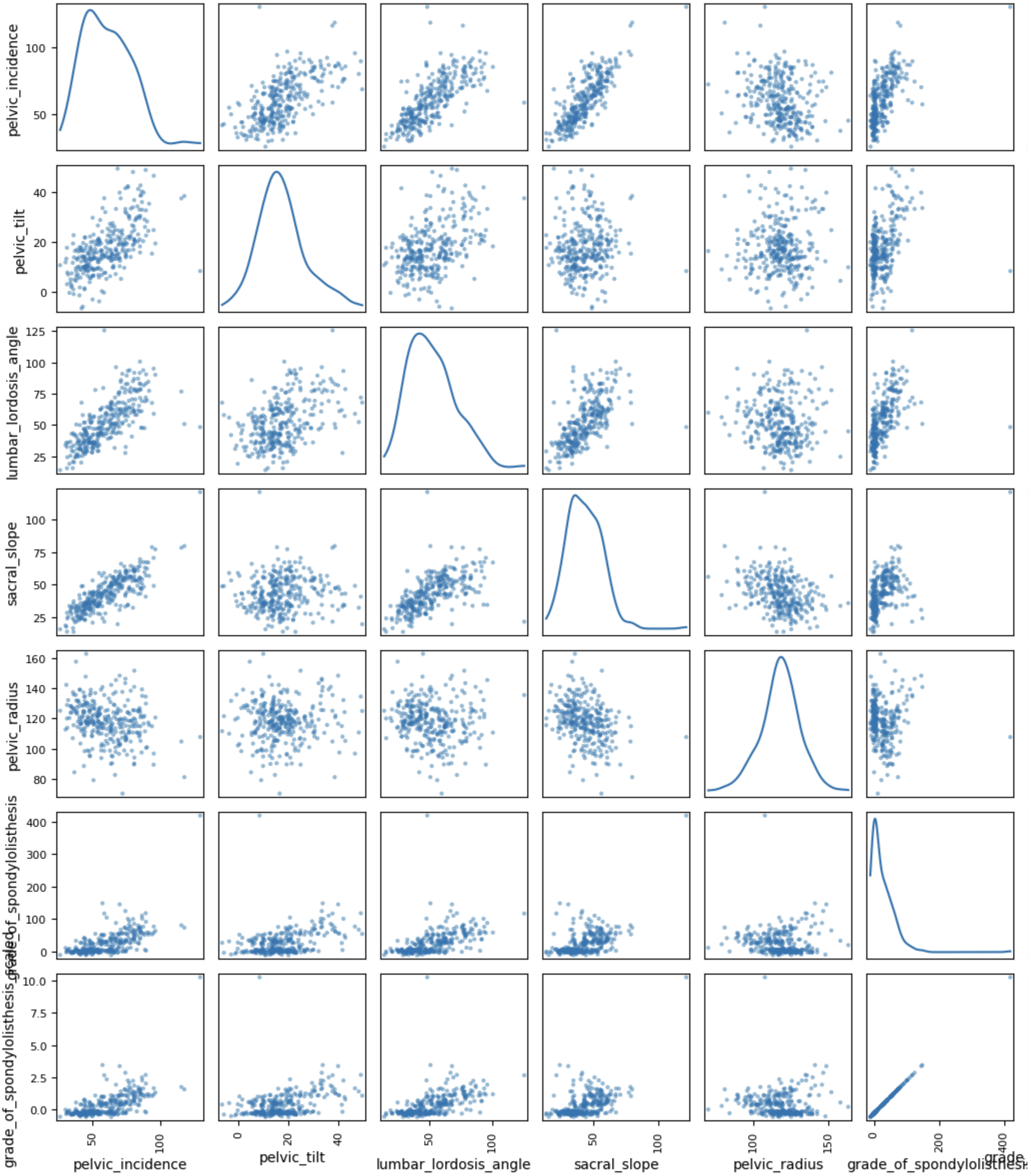


In [3]:
```
#box plot
data.boxplot(by='class', figsize=(10, 10), layout=(3,2))
plt.tight_layout()
plt.show()
```



In [102…

```
# Scatter plots
pd.plotting.scatter_matrix(data.drop('class', axis=1), figsize=(12, 12), diagonal='kde')
plt.tight_layout()
plt.show()
```



Data Preprocessing

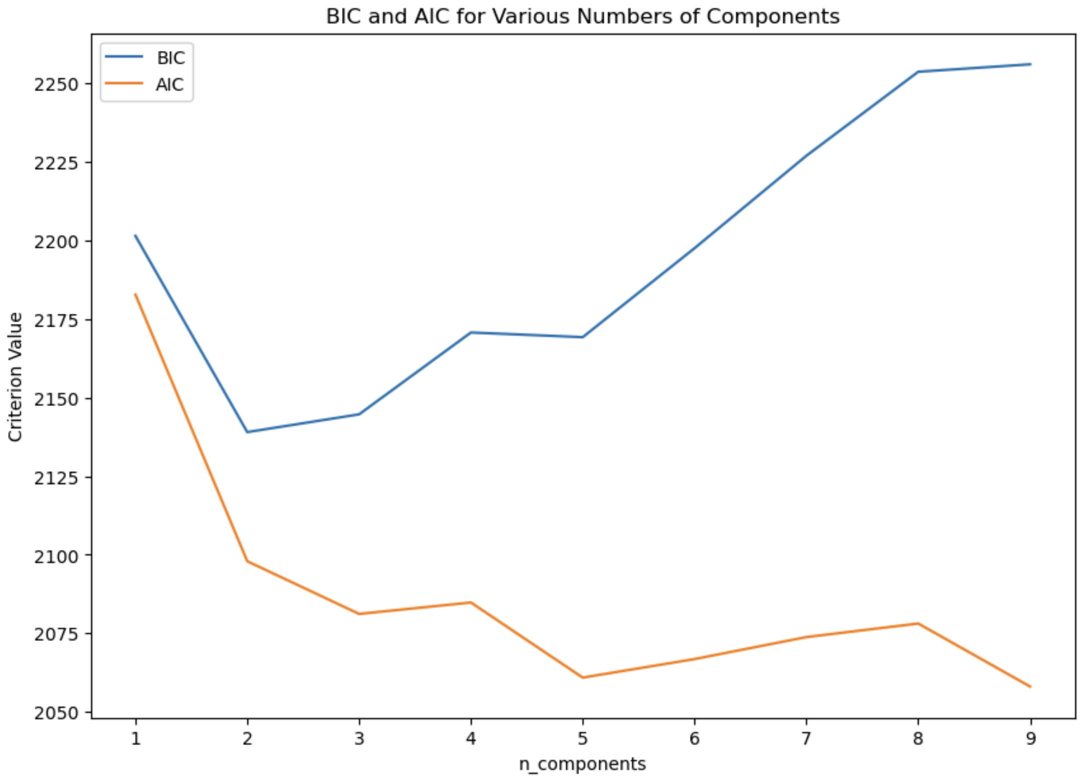
In [10]:
```
numerical_features = data.drop(['class'], axis=1)

# Outlier handling for 'grade_of_spondylolisthesis'
robust_scaler = RobustScaler()
data['grade_of_spondylolisthesis_scaled'] = robust_scaler.fit_transform(data[['grade_of_spondylolisthesis']])
numerical_features = numerical_features.drop(['grade_of_spondylolisthesis'], axis=1)

# StandardScaler to the rest of the numerical features
standard_scaler = StandardScaler()
scaled_features = standard_scaler.fit_transform(numerical_features)

# robust scaled 'grade_of_spondylolisthesis'+ other scaled features
scaled_data = pd.DataFrame(scaled_features, columns=numerical_features.columns)
scaled_data['grade_of_spondylolisthesis_scaled'] = data['grade_of_spondylolisthesis_scaled']

# PCA
pca = PCA(n_components=2)
principalComponents = pca.fit_transform(scaled_data)

# df
pcadf = pd.DataFrame(data = principalComponents, columns = ['PC1', 'PC2'])

pcadf.head()
```

Out[10]:

	PC1	PC2
0	-0.198703	-0.872628
1	-2.242583	-0.420751
2	0.327399	-0.624497
3	0.404378	-0.636299
4	-1.361614	-1.072244

Model-based: Gausian Mixture Model

In [33]:
```
pcadf_numerical_only = pcadf[['PC1', 'PC2']]

n_components = np.arange(1, 10)

models = [GaussianMixture(n, covariance_type='full', random_state=0).fit(pcadf_numerical_only)
          for n in n_components]

plt.figure(figsize=(10, 7))
plt.plot(n_components, [m.bic(pcadf_numerical_only) for m in models], label='BIC')
plt.plot(n_components, [m.aic(pcadf_numerical_only) for m in models], label='AIC')
plt.legend(loc='best')
plt.xlabel('n_components')
plt.ylabel('Criterion Value')
plt.title('BIC and AIC for Various Numbers of Components')
plt.show()
```



GMM for k=2

In [22]:
```
gmm = GaussianMixture(n_components=2, random_state=42)
gmm.fit(principalComponents)
clusters = gmm.predict(principalComponents)

# Add cluster assignments back to the original data
data['cluster'] = clusters

# Display the first few rows of the updated dataset
data.head()
```

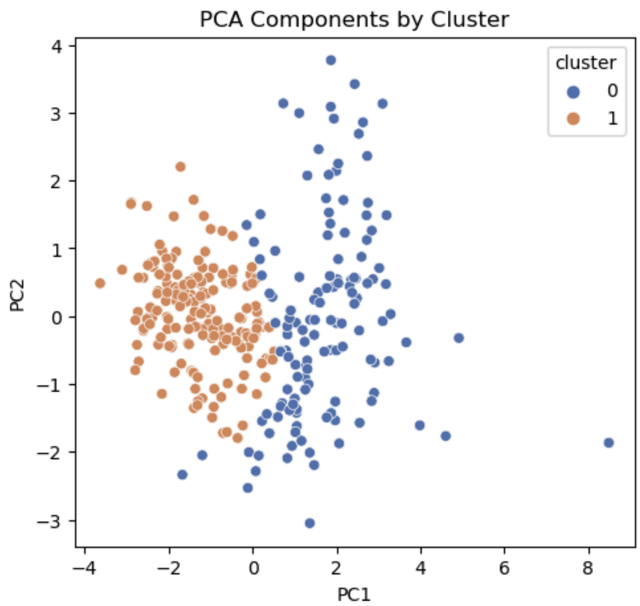
Out[22]:

	pelvic_incidence	pelvic_tilt	lumbar_lordosis_angle	sacral_slope	pelvic_radius	grade_of_spondylolisthesis	class	grade_of_spondylolisthesis_scaled	cluster
0	63.03	0.540493	39.61	40.48	98.67	-0.302759	AB	-0.302759	1
1	39.06	-0.550098	25.02	29.00	114.41	-0.181555	AB	-0.181555	1
2	68.83	0.511679	50.09	46.61	105.99	-0.385410	AB	-0.385410	1
3	69.30	0.723859	44.31	44.64	101.87	-0.013985	AB	-0.013985	1
4	49.71	-0.585898	28.32	40.06	108.17	-0.096888	AB	-0.096888	1

In [29]:
```
## Evaluating GMM clustering

# Mapping clusters to labels
def map_clusters_to_labels(cluster_labels, true_labels):
    label_mapping = {}
    for cluster in np.unique(cluster_labels):
        # Find the most frequent true label in each cluster
        mode_label = true_labels[cluster_labels == cluster].mode()[0]
        label_mapping[cluster] = mode_label
    return label_mapping

# Map predicted clusters to actual labels
label_mapping = map_clusters_to_labels(data['cluster'], data['class'])
mapped_labels = data['cluster'].map(label_mapping)

# Evaluate clustering
accuracy = accuracy_score(data['class'], mapped_labels)
print("Accuracy score is", accuracy)
conf_matrix = confusion_matrix(data['class'], mapped_labels, labels=["AB", "NO"])
print("Confusion matrix:", conf_matrix)

# Visualisation
# PCA Components with cluster assignment
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
sns.scatterplot(x="PC1", y="PC2", hue=data['cluster'], palette="deep", data=pcadf)
plt.title('PCA Components by Cluster')
```

```
Accuracy score is 0.6774193548387096
Confusion matrix: [[210   0]
 [100   0]]
```

Out[29]:
```
Text(0.5, 1.0, 'PCA Components by Cluster')
```



SVM

In [42]:
```
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)

# Split train-test
X_train_pca, X_test_pca, y_train, y_test = train_test_split(X_pca, y, test_size=0.3, random_state=42)

# Train model
svm_model_pca = SVC(kernel='linear', random_state=42)
svm_model_pca.fit(X_train_pca, y_train)

# Make Predictions on test
y_pred_pca = svm_model_pca.predict(X_test_pca)
```

In [45]:
```
#evaluation

accuracy_pca = accuracy_score(y_test, y_pred_pca)
print("Accuracy score with PCA is", accuracy_pca)

conf_matrix_pca = confusion_matrix(y_test, y_pred_pca)
print("Confusion matrix with PCA:\n", conf_matrix_pca)

report_pca = classification_report(y_test, y_pred_pca)
print("Classification report with PCA:\n", report_pca)
```

```
Accuracy score with PCA is 0.7204301075268817
Confusion matrix with PCA:
 [[14 10]
 [16 53]]
Classification report with PCA:
               precision    recall  f1-score   support

           0       0.47      0.58      0.52        24
           1       0.84      0.77      0.80        69

    accuracy                           0.72        93
   macro avg       0.65      0.68      0.66        93
weighted avg       0.74      0.72      0.73        93
```