STATISTICS AND MACHINE LEARNING 2

ASSIGNMENT 1

Student ID: 11340063

Word Count:1093

K-means clustering unsupervised algorithm:

K-means clustering was employed for unsupervised learning. The technique divides a dataset into K unique, non-overlapping clusters iteratively [1]. The aim is to combine comparable data points while reducing the variance within the cluster. K cluster centroids are initially selected at random. The cluster whose centroid is closest to each data point is assigned to it. The closest centroid is assigned to each data point after the distance to each centroid is computed. Euclidean distance is usually used in this case given by the formula:

$$D(x_i, c_j) = \sqrt{\sum_{k=1}^{d}(x_{ik} - c_{jk})^2}$$

Where x_{ik} is the k^{th} dimension of data point x_i and c_{jk} is the k^{th} dimension of centroid c_j.

The new centroid of the cluster is recalculated by taking the mean of all data points assigned to that cluster. The new c_j' for cluster j is computed as :

$$c_j' = \frac{1}{N_j}\sum_{i=1}^{N_j} x_i$$

Where N_j is the number of data points in cluster j. Until the distribution of data points among clusters is either completely stable or barely fluctuates, these steps are repeated. Finding centroids that minimise the within-cluster sum of squares (WCSS), or the total of the squared distances between each data point and its designated centroid within the cluster, is the goal of the K-means algorithm.

$$WCSS = \sum_{j=1}^{K}\sum_{i=1}^{N_j} \|x_i - c_j\|^2$$

Here N_j is the number of data points in cluster j, x_i is a data point and c_j is the centroid of cluster j.

K-Nearest Neighbour supervised algorithm:

A straightforward and understandable classification method is the K-Nearest Neighbours (KNN) algorithm. A data point is classified according to the feature space's K nearest neighbours' majority class [2]. A labelled training dataset with a feature vector and matching class label is fed into the model to be trained. For prediction, we consider each data point on the test dataset as x_{new} that is to be classified. The Euclidean distance between x_{new} and every data point in training is calculated.

$$D(x_{\text{new}}, x_i) = \sqrt{\sum_{k=1}^{d}(x_{\text{new},k} - x_{i,k})^2}$$

Where x_{ik} is each data point in the k^{th} cluster in training data.

Based on the distances calculated, the K nearest neighbours of x_{new} are identified. Then x_{new} is assigned to the class that is most frequent among its K neighbours.

$$C_{\text{pred}} = \arg\max_k N_k$$

Here C_{pred} is the final predicted class and N_k is the class of the neighbour in k^{th} cluster.

Exploratory Data Analysis and Data Preprocessing:

The dataset used in this coursework includes six biomechanical characteristics that are used to categorise orthopaedic patients into two classes: Normal (NO) and Abnormal (AB). After closely examining the dataset, it appears that there are no missing values. Two of the six features—the Pelvic Tilt and the Grade of Spondylolisthesis—have minimum negative values. It is evident from Figure 1 that the dataset has over 200 records that describe the class Abnormal.

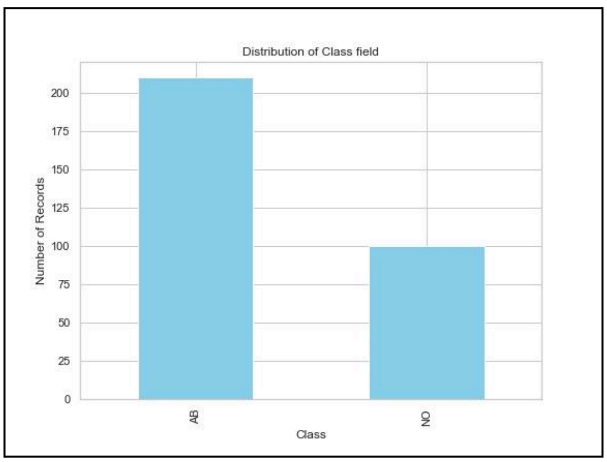


Figure 1: Distribution of records in Class field

Though the dataset is free from missing values, from Figure 2 it can be seen that the dataset contains outliers.

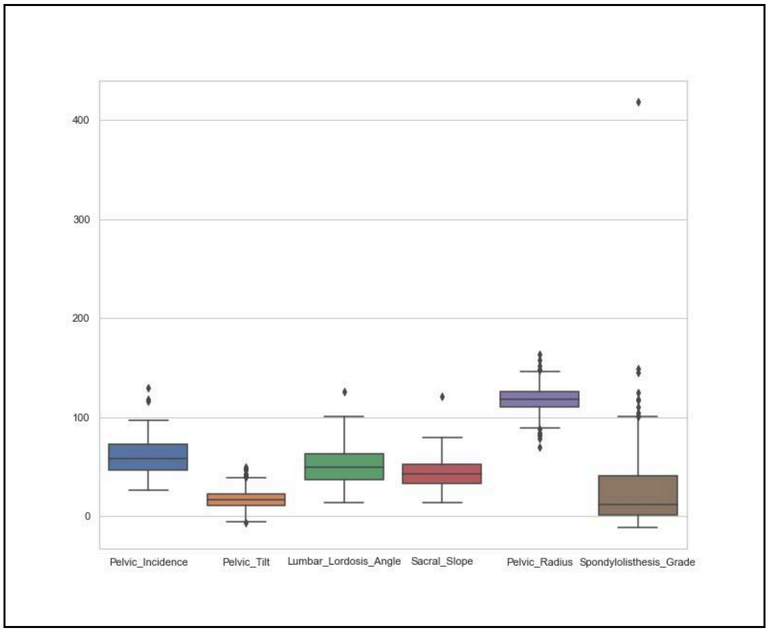


Figure 2: Boxplot of the dataset

Upon closer inspection, Figure 3 reveals that the majority of these extreme values within each feature belong to the Abnormal class. This may help to clarify why the orthopaedic patients in question were placed in the Abnormal category.

```
Number of outliers in each column:
Pelvic_Incidence           3
Pelvic_Tilt               13
Lumbar_Lordosis_Angle      1
Sacral_Slope               1
Pelvic_Radius             11
Spondylolisthesis_Grade   10
dtype: int64


Column names containing outliers and their corresponding values in 'Class' column:
+-------------------------+------------------------------------------------------------------+
|         Column          |                              Values                              |
+-------------------------+------------------------------------------------------------------+
|    Pelvic_Incidence     |                        ['AB' 'AB' 'AB']                          |
|       Pelvic_Tilt       | ['AB' 'AB' 'AB' 'AB' 'AB' 'AB' 'AB' 'AB' 'AB' 'AB' 'AB' 'AB' 'AB'] |
|  Lumbar_Lordosis_Angle  |                              ['AB']                              |
|      Sacral_Slope       |                              ['AB']                              |
|      Pelvic_Radius      |      ['AB' 'AB' 'AB' 'AB' 'AB' 'AB' 'AB' 'AB' 'AB' 'AB' 'NO']      |
| Spondylolisthesis_Grade |        ['AB' 'AB' 'AB' 'AB' 'AB' 'AB' 'AB' 'AB' 'AB' 'AB']         |
+-------------------------+------------------------------------------------------------------+
```

Figure 3: Explanation for outliers

Figure 4 displays the dataset's correlation heatmap. Pelvic_Tilt and Pelvic_Incidence have the strongest positive correlation, followed by Lumbar_Lordosis_Angle and Pelvic_Incidence, according to the figure. Maximum negative correlation is observed between Sacral_Slope and Pelvic_Radius.

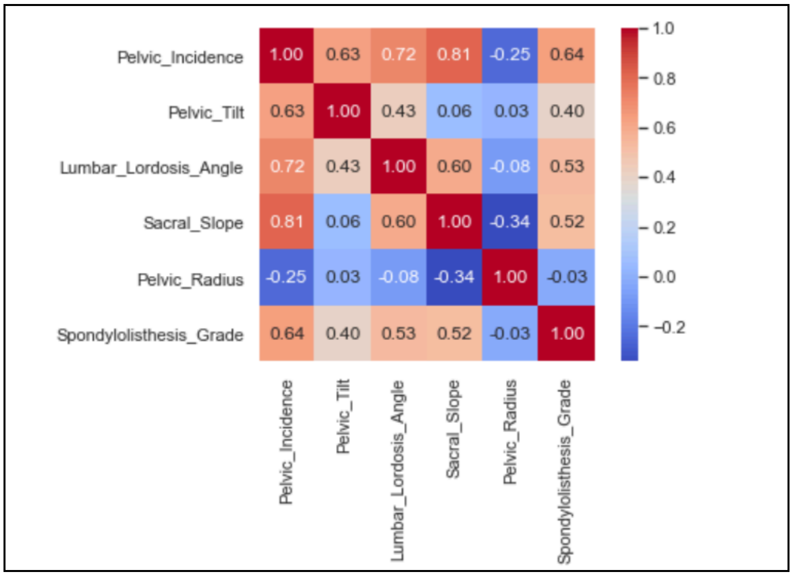


Figure 4: Correlation Heatmap of the dataset

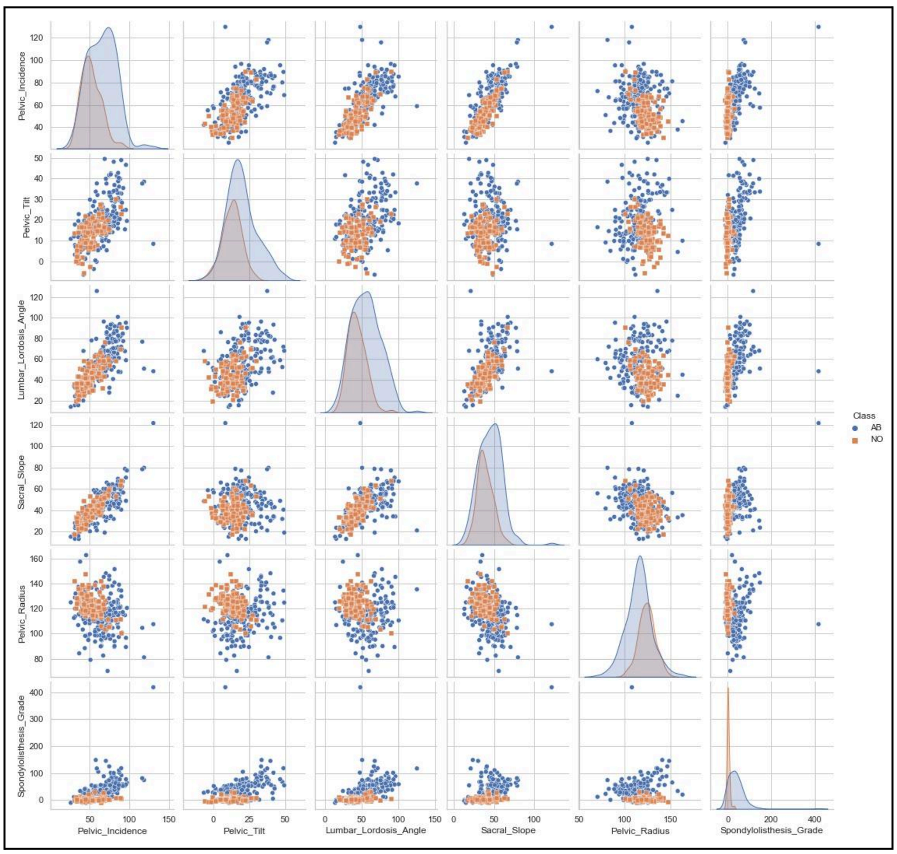


Figure 5: Pairplot of the dataset

To view the data point distribution, a pairplot was created. Figure 4 illustrates the skewness of the data points. It is also observed that the clusters are heavily overlapped. This can complicate both supervised and unsupervised model clustering and classification. The logarithmic transformation was used to distribute the data points in order to solve this problem [3]. The dataset was shifted by a constant of 12 because it contains negative points, and it was then logarithmically transformed. In order to take into account the smallest number from every column, which was almost -11.06, the number 12 was selected. Figure 6 displays the transformed data pairplot. The figure makes it evident that data points were altered to eliminate cluster overlap. By this transformation, the dataset's skewness is also fixed.

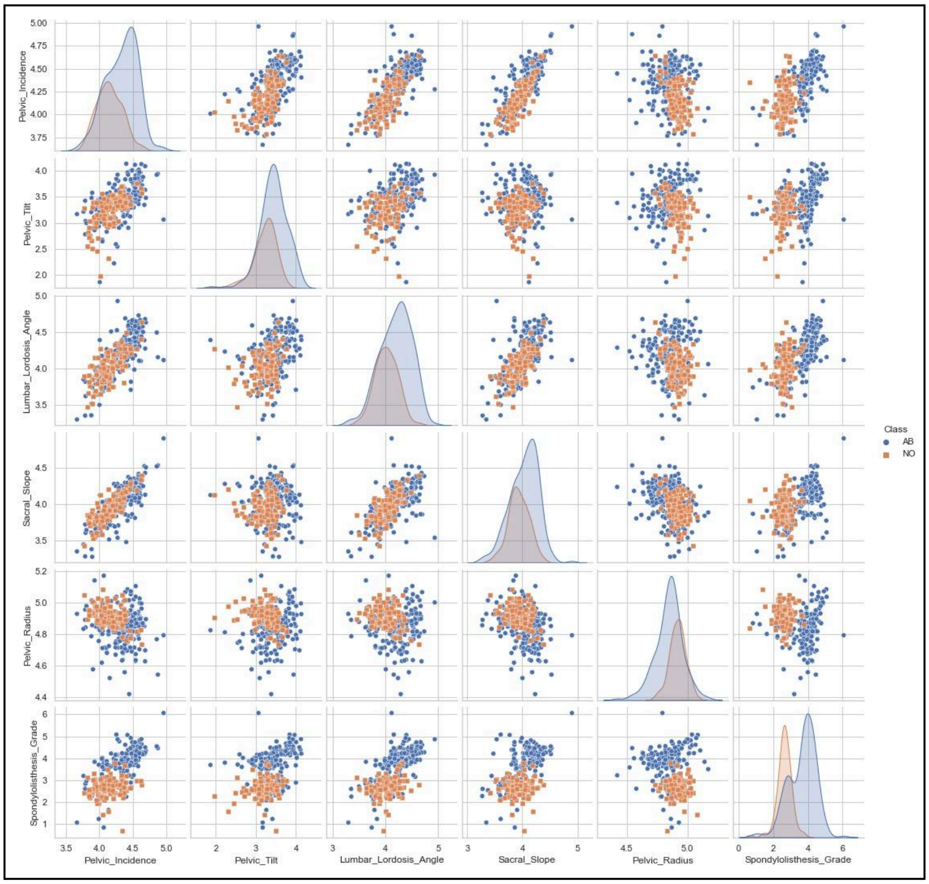


Figure 6: Pairplot of the dataset after logarithmic transformation

The Class column of the transformed data is then Label Encoded in such a way that 0 represents AB and 1 represents NO.

Analysis and Results:

To achieve the best results when using the K-means clustering model, it is imperative to make sure the dataset has been properly preprocessed. Reducing the dimensionality of the data helps to improve the performance of clustering. For this, the Principal Component Analysis (PCA) method was employed [4]. PCA is helpful when working with high-dimensional datasets because it captures the important information in a reduced-dimensional space. An elbow plot was created in order to ascertain the ideal number of clusters (k) for the K-means algorithm [5]. According to the plot in Figure 7, the "elbow" or point of diminishing returns happened at k=2. This number denotes the ideal number of clusters for the dataset, striking a compromise between preserving the data's useful structure and avoiding undue granularity. Thus, a k value of 2 was chosen for the subsequent application of K-means clustering.

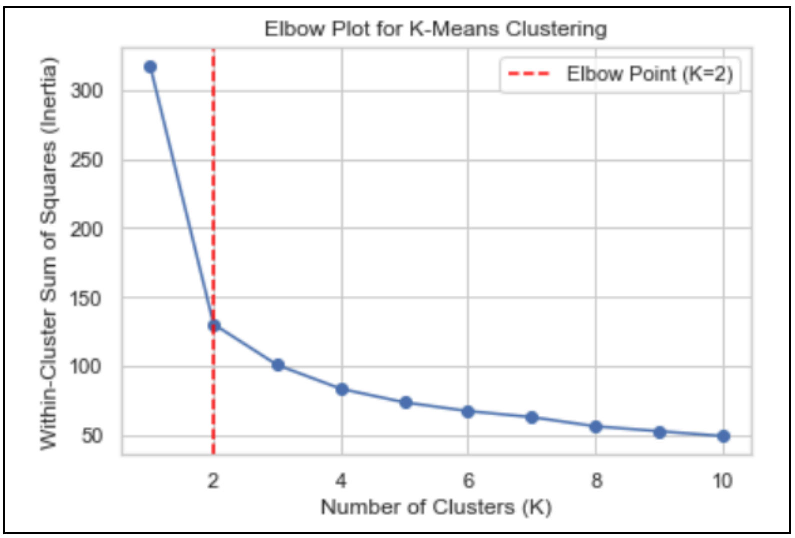


Figure 7: Elbow plot for K-Means clustering

The transformed dataset is then passed on to PCA for dimensionality reduction. The dataset is reduced to 2 principal components for easier visualisation. After clustering the Adjusted Rand Index score was calculated, which was found to be 0.3352. The resultant clustering can be seen in Figure 8.

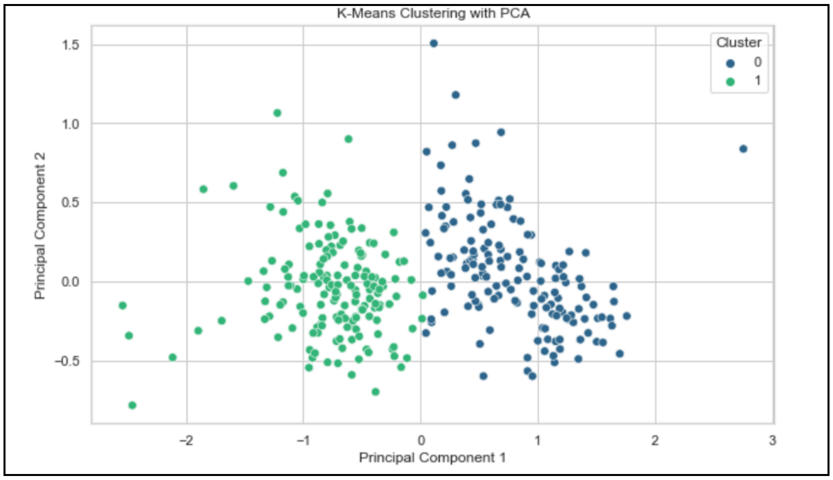


Figure 8: K means clustering with PCA

It was determined not to use Principal Component Analysis (PCA) in the context of K-nearest neighbours (KNN) classification, in contrast to its use for K-means clustering. The decision was motivated by the dataset's low dimensionality, with only six features. In these situations, keeping the original features may be essential to preserving data relevant to classification tasks.

After that, 20% of the dataset was set aside for testing in order to assess the model's performance on untested data. Using the training dataset, a cross-validation process was used to find the ideal value for the KNN parameter k [6]. 9 was determined to be the most useful value for k after this investigation.

This ideal 'k' value was then used to train the KNN model, yielding an accuracy of 81%. A metric called the F1-score, which takes into account both precision and recall, was also computed. A balanced trade-off between recall and precision was evident in the obtained F1-score of 67% in the classification task.

References:

[1] Na, S., Xumin, L. and Yong, G. (2010). Research on k-means Clustering Algorithm: An Improved k-means Clustering Algorithm. 2010 Third International Symposium on Intelligent Information Technology and Security Informatics. doi:https://doi.org/10.1109/iitsi.2010.74.

[2] Guo, G., Wang, H., Bell, D., Bi, Y. and Greer, K. (2003). KNN Model-Based Approach in Classification. On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE, 2888, pp.986–996. doi:https://doi.org/10.1007/978-3-540-39964-3_62.

[3] Feng, C., Wang, H., Lu, N., Chen, T., He, H., Lu, Y. and Tu, X. (2014). Log-transformation and its implications for data analysis. Shanghai archives of psychiatry, [online] 26(2), pp.105–109. doi:https://doi.org/10.3969/j.issn.1002-0829.2014.02.009.

[4] Maćkiewicz, A. and Ratajczak, W. (1993). Principal components analysis (PCA). Computers & Geosciences, 19(3), pp.303–342. doi:https://doi.org/10.1016/0098-3004(93)90090-r.

[5] Syakur, M.A., Khotimah, B.K., Rochman, E.M.S. and Satoto, B.D. (2018). Integration K-Means Clustering Method and Elbow Method For Identification of The Best Customer Profile Cluster. IOP Conference Series: Materials Science and Engineering, 336, p.012017. doi:https://doi.org/10.1088/1757-899x/336/1/012017.

[6] Schaffer, C. (1993). Selecting a classification method by cross-validation. Machine Learning, 13(1), pp.135–143. doi:https://doi.org/10.1007/bf00993106.

Appendix

```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score
from sklearn.cluster import KMeans
from sklearn.metrics import adjusted_rand_score
from sklearn.model_selection import cross_val_score
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from prettytable import PrettyTable
```

```
col=['Pelvic_Incidence','Pelvic_Tilt',
'Lumbar_Lordosis_Angle','Sacral_Slope','Pelvic_Radius','Spondylolisthe
sis_Grade','Class']
df=pd.read_csv('dataset.txt',delimiter=' ', header=None, names=col)
```

```
df
```

```
     Pelvic_Incidence  Pelvic_Tilt  Lumbar_Lordosis_Angle
Sacral_Slope  \
0               63.03        22.55                  39.61
40.48
1               39.06        10.06                  25.02
29.00
2               68.83        22.22                  50.09
46.61
3               69.30        24.65                  44.31
44.64
4               49.71         9.65                  28.32
40.06
..                ...          ...                    ...           ..
.
305             47.90        13.62                  36.00
34.29
306             53.94        20.72                  29.22
33.22
307             61.45        22.69                  46.17
38.75
308             45.25         8.69                  41.58
36.56
```

```
309              33.84        5.07                      36.64
28.77

     Pelvic_Radius  Spondylolisthesis_Grade Class
0            98.67                    -0.25    AB
1           114.41                     4.56    AB
2           105.99                    -3.53    AB
3           101.87                    11.21    AB
4           108.17                     7.92    AB
..             ...                      ...   ...
305         117.45                    -4.25    NO
306         114.37                    -0.42    NO
307         125.67                    -2.71    NO
308         118.55                     0.21    NO
309         123.95                    -0.20    NO

[310 rows x 7 columns]
```

```
df.info()
```

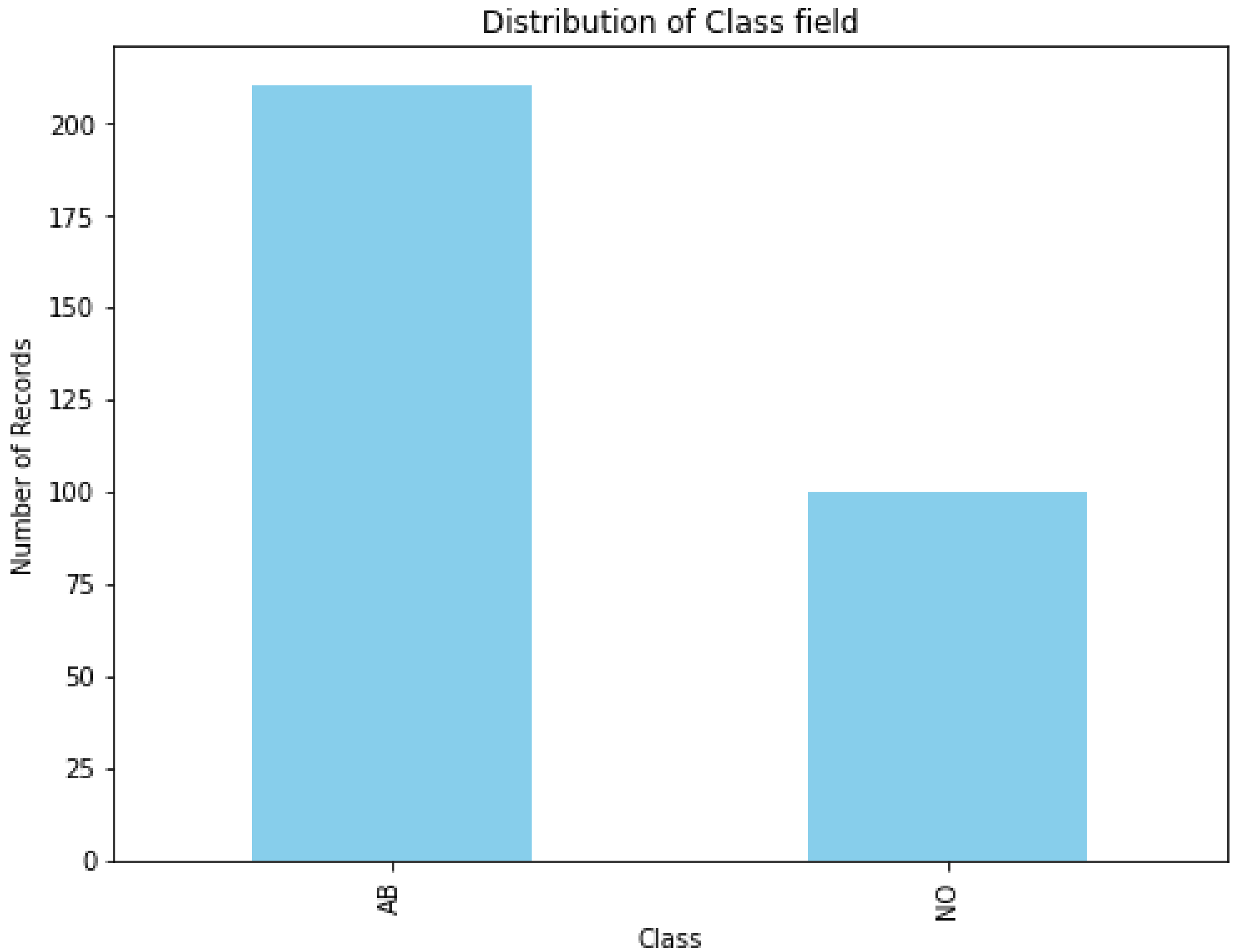
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 310 entries, 0 to 309
Data columns (total 7 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pelvic_Incidence          310 non-null    float64
 1   Pelvic_Tilt               310 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   Spondylolisthesis_Grade   310 non-null    float64
 6   Class                     310 non-null    object
dtypes: float64(6), object(1)
memory usage: 17.1+ KB
```

```
df.describe()
```

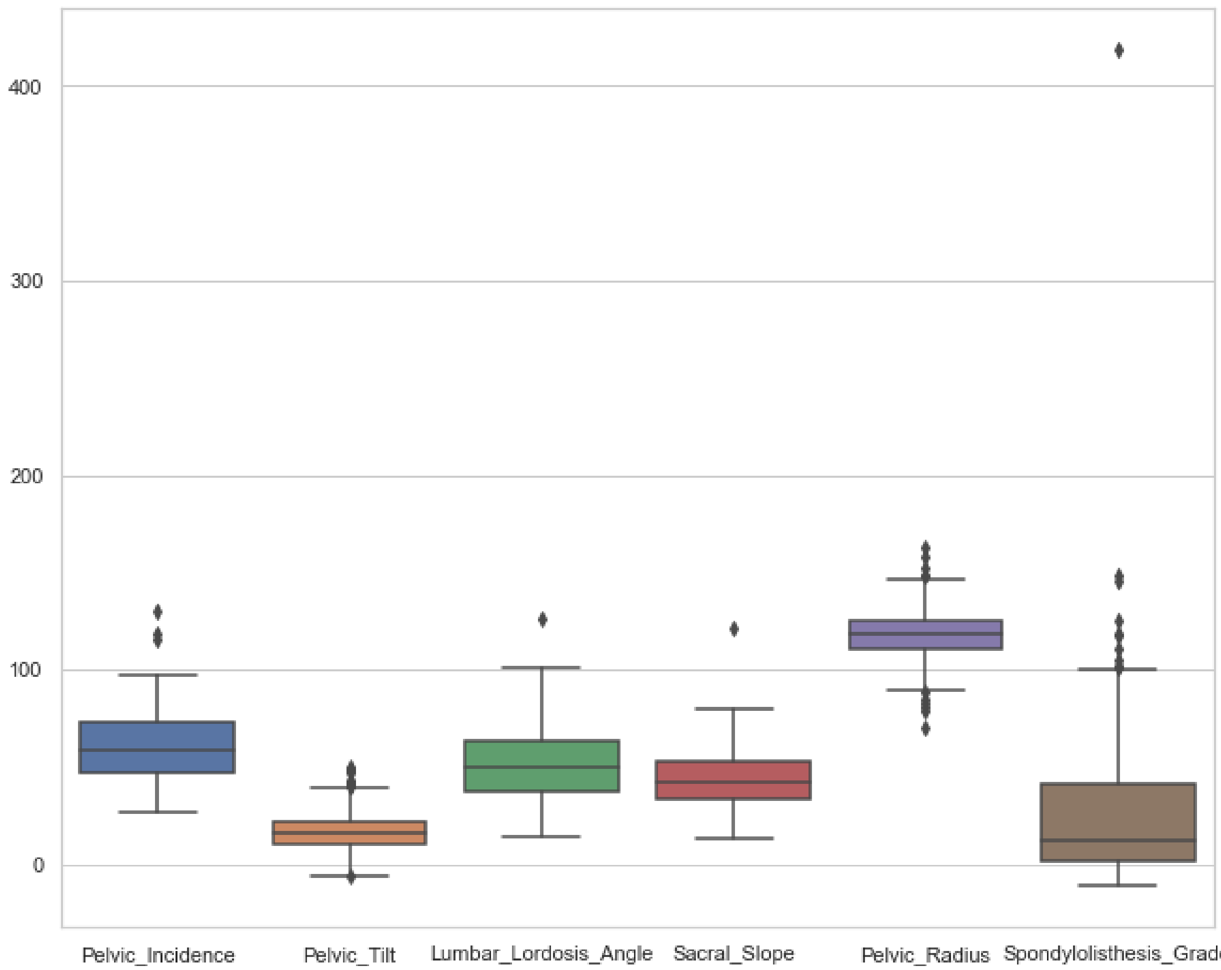
```
       Pelvic_Incidence  Pelvic_Tilt  Lumbar_Lordosis_Angle
Sacral_Slope  \
count        310.000000   310.000000             310.000000
310.000000
mean          60.496484    17.542903              51.930710
42.953871
std           17.236109    10.008140              18.553766
13.422748
min           26.150000    -6.550000              14.000000
13.370000
25%           46.432500    10.667500              37.000000
33.347500
50%           58.690000    16.360000              49.565000
```

```
42.405000
75%              72.880000    22.120000              63.000000
52.692500
max             129.830000    49.430000             125.740000
121.430000

       Pelvic_Radius  Spondylolisthesis_Grade
count     310.000000               310.000000
mean      117.920548                26.296742
std        13.317629                37.558883
min        70.080000               -11.060000
25%       110.710000                 1.600000
50%       118.265000                11.765000
75%       125.467500                41.285000
max       163.070000               418.540000
```

```
class_counts = df['Class'].value_counts()

# Plotting a histogram
plt.figure(figsize=(8, 6))
class_counts.plot(kind='bar', color='skyblue')
plt.title('Distribution of Class field')
plt.xlabel('Class')
plt.ylabel('Number of Records')

plt.savefig('class_distribution.jpg')

plt.show()
```




```
sns.set(style="whitegrid")
plt.figure(figsize=(11, 9))
sns.boxplot(data=df[col])

plt.savefig('boxplot.jpg')
plt.show()
```

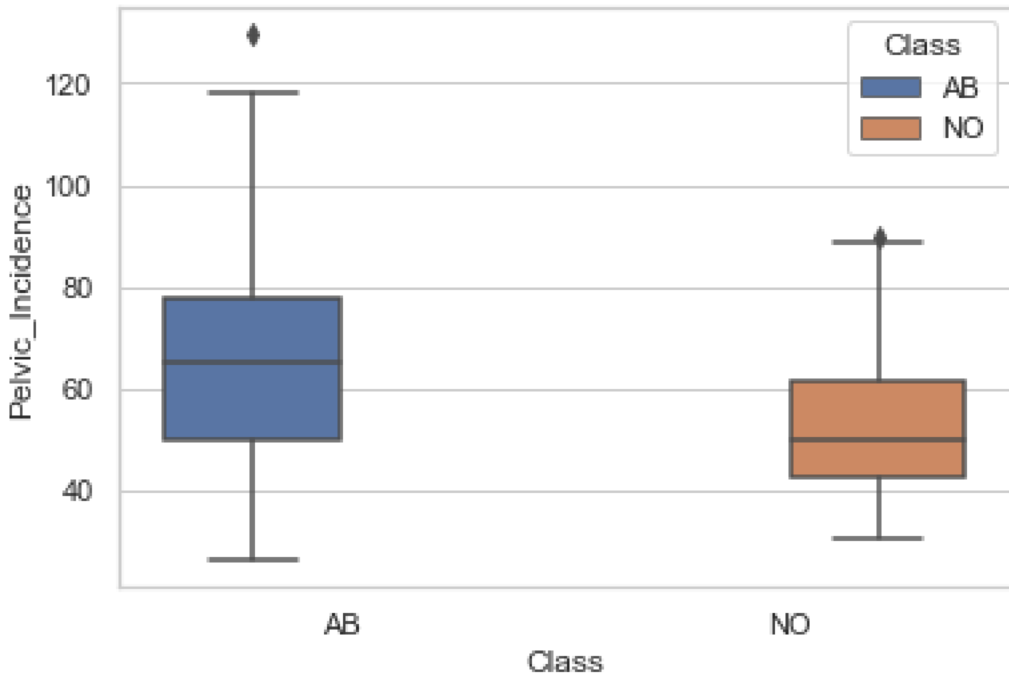



```
target_column = df['Class']

# Identifying no of outliers in each column
numeric_col = df.select_dtypes(include=['float64', 'int64'])
Q1 = numeric_col.quantile(0.25)
Q3 = numeric_col.quantile(0.75)
IQR = Q3 - Q1
outliers_mask = ((numeric_col < (Q1 - 1.5 * IQR)) | (numeric_col > (Q3
+ 1.5 * IQR)))

print("Number of outliers in each column:")
print(outliers_mask.sum())

# Finding the corresponding class label for each outlier in each
column
outliers_info = []
for column in numeric_col.columns:
    column_outliers = target_column[outliers_mask[column]]
    if not column_outliers.empty:
        outliers_info.append((column, column_outliers.values))
```

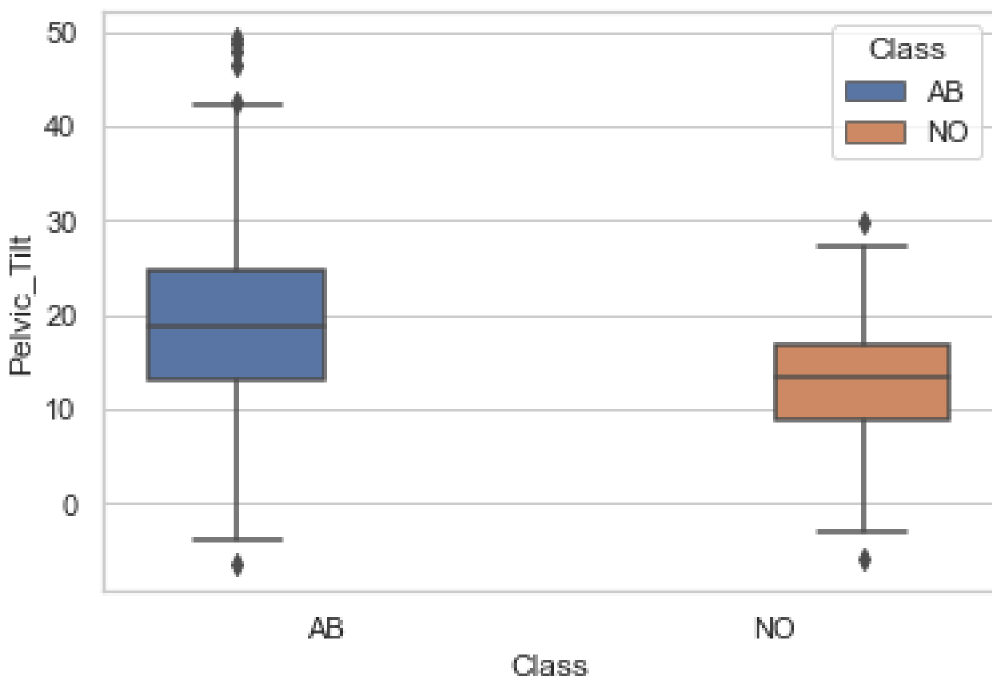
```python
# Display the column names and values of 'class' column corresponding to outliers
print("\n\nColumn names containing outliers and their corresponding values in 'Class' column:")
table = PrettyTable()
table.field_names = ['Column', 'Values']

for column, values in outliers_info:
    table.add_row([column, values])

print(table)
```

```
Number of outliers in each column:
Pelvic_Incidence            3
Pelvic_Tilt                13
Lumbar_Lordosis_Angle       1
Sacral_Slope                1
Pelvic_Radius              11
Spondylolisthesis_Grade    10
dtype: int64


Column names containing outliers and their corresponding values in 'Class' column:
+-------------------------
+-----------------------------------------------------------------+
|          Column         |                              Values
|
+-------------------------
+-----------------------------------------------------------------+
|     Pelvic_Incidence    |                         ['AB' 'AB' 'AB']
|
|       Pelvic_Tilt       | ['AB' 'AB' 'AB' 'AB' 'AB' 'AB' 'AB' 'AB'
'AB' 'AB' 'AB' 'AB' 'AB'] |
|  Lumbar_Lordosis_Angle  |                              ['AB']
|
|       Sacral_Slope      |                              ['AB']
|
|       Pelvic_Radius     |     ['AB' 'AB' 'AB' 'AB' 'AB' 'AB' 'AB'
'AB' 'AB' 'AB' 'NO']       |
| Spondylolisthesis_Grade |       ['AB' 'AB' 'AB' 'AB' 'AB' 'AB' 'AB'
'AB' 'AB' 'AB']         |
+-------------------------
+-----------------------------------------------------------------+
```

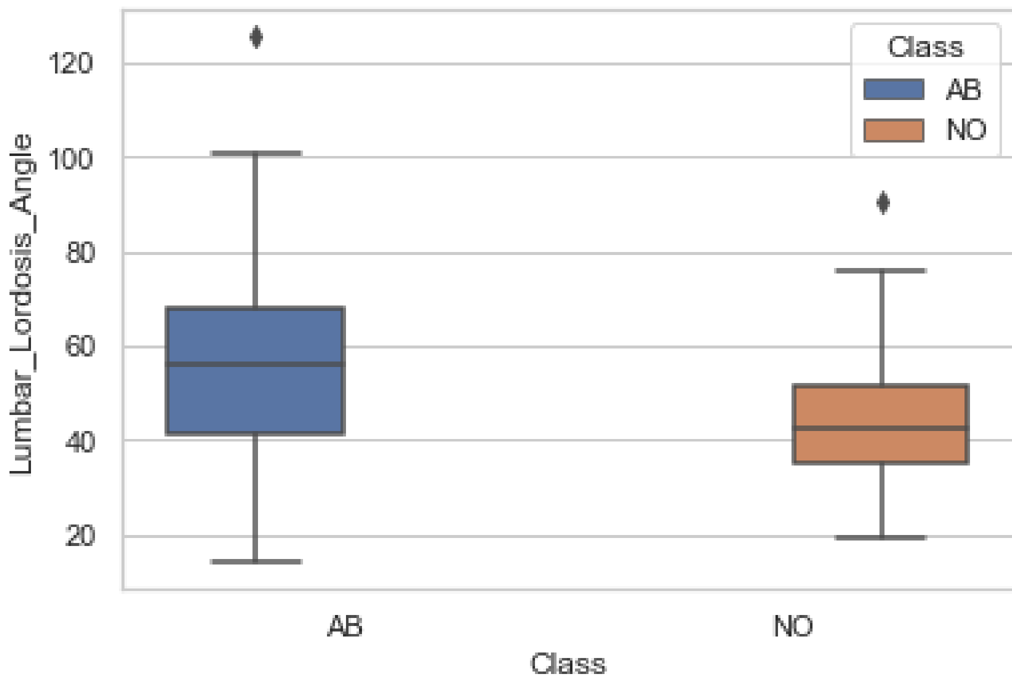
```python
sns.boxplot(x = 'Class', y = 'Pelvic_Incidence', data = df, hue =
'Class')
plt.show()
```



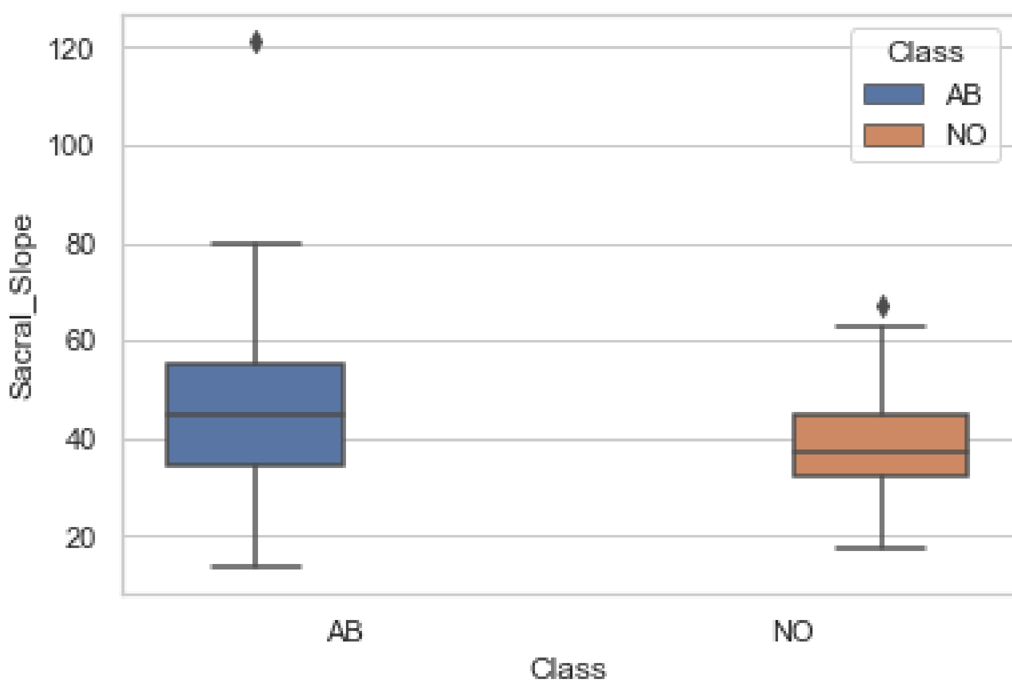

```
sns.boxplot(x = 'Class', y = 'Pelvic_Tilt', data =df, hue = 'Class')
plt.show()
```



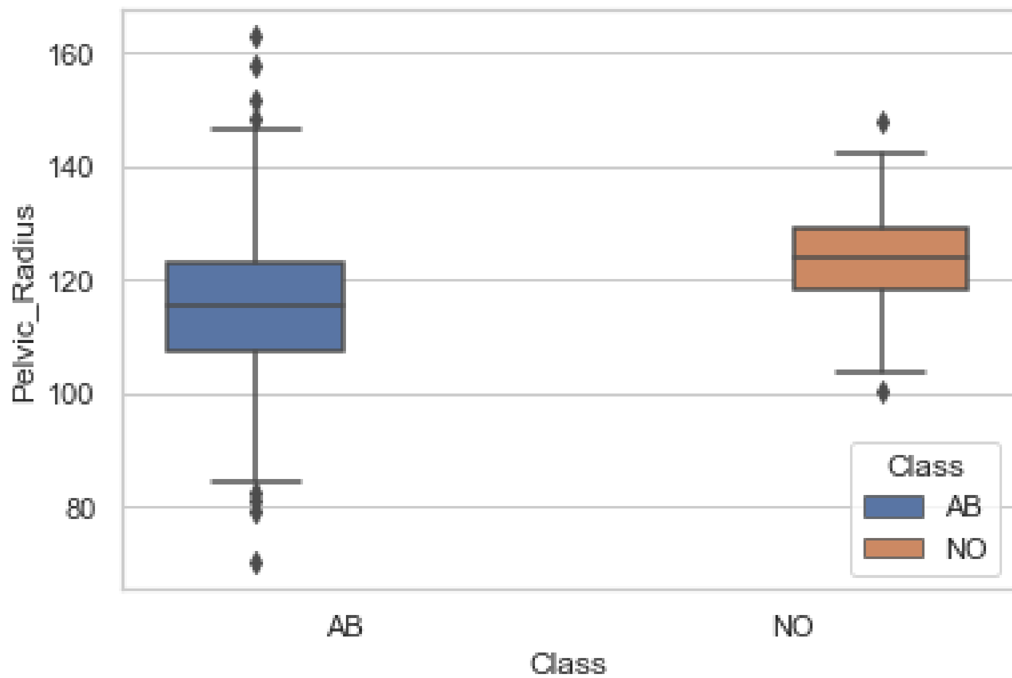

```
sns.boxplot(x = 'Class', y = 'Lumbar_Lordosis_Angle', data = df, hue =
'Class')
plt.show()
```



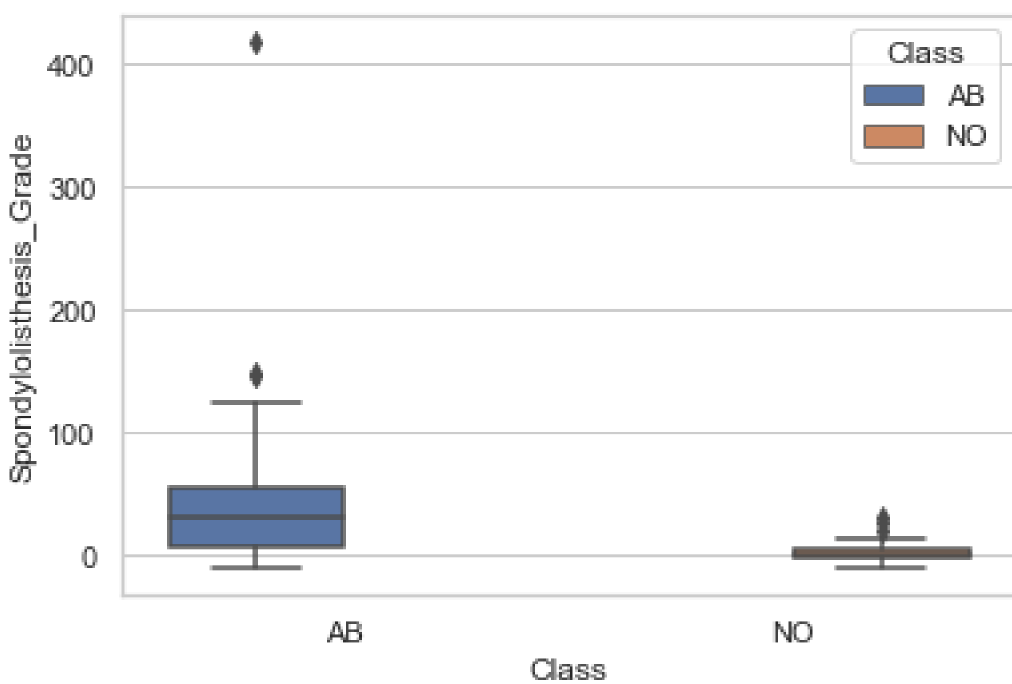

```
sns.boxplot(x = 'Class', y = 'Sacral_Slope', data = df, hue = 'Class')
plt.show()
```




```
sns.boxplot(x = 'Class', y = 'Pelvic_Radius', data = df, hue =
'Class')
plt.show()
```

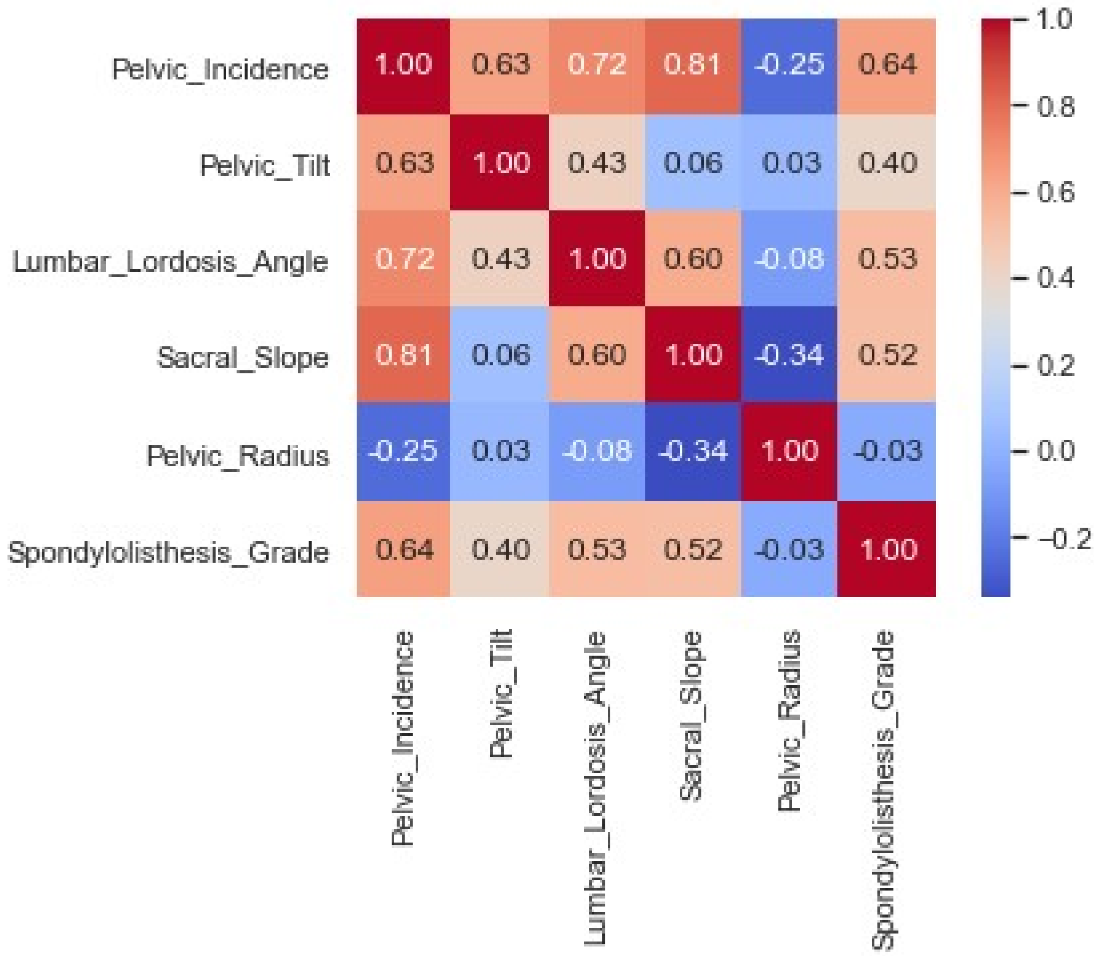



```
sns.boxplot(x = 'Class', y = 'Spondylolisthesis_Grade', data = df, hue
= 'Class')
plt.show()
```

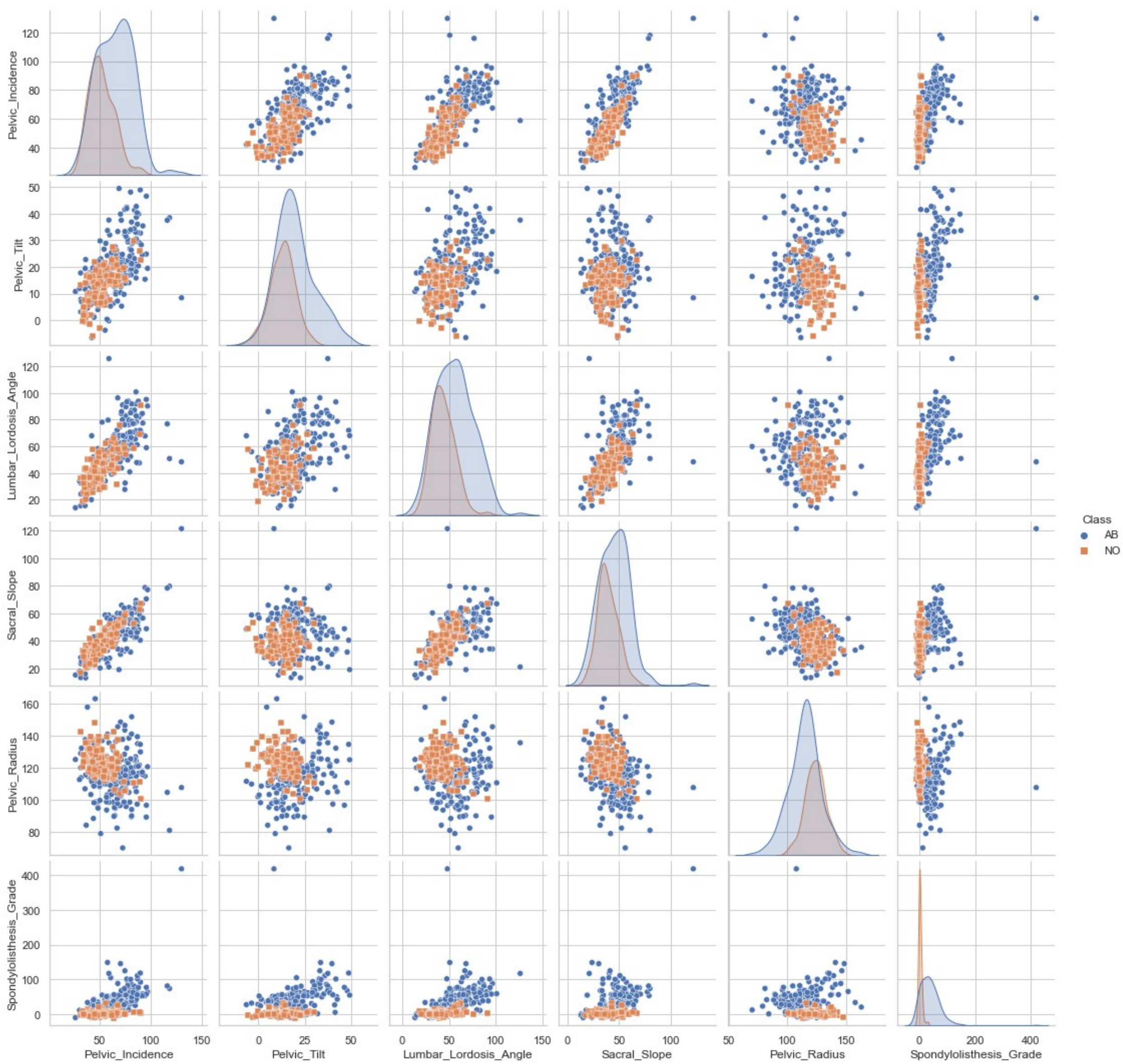



```
# Correlation Matrix
correlation_matrix = df.corr()
```

```
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm",
fmt=".2f", square=True)
plt.show()
```



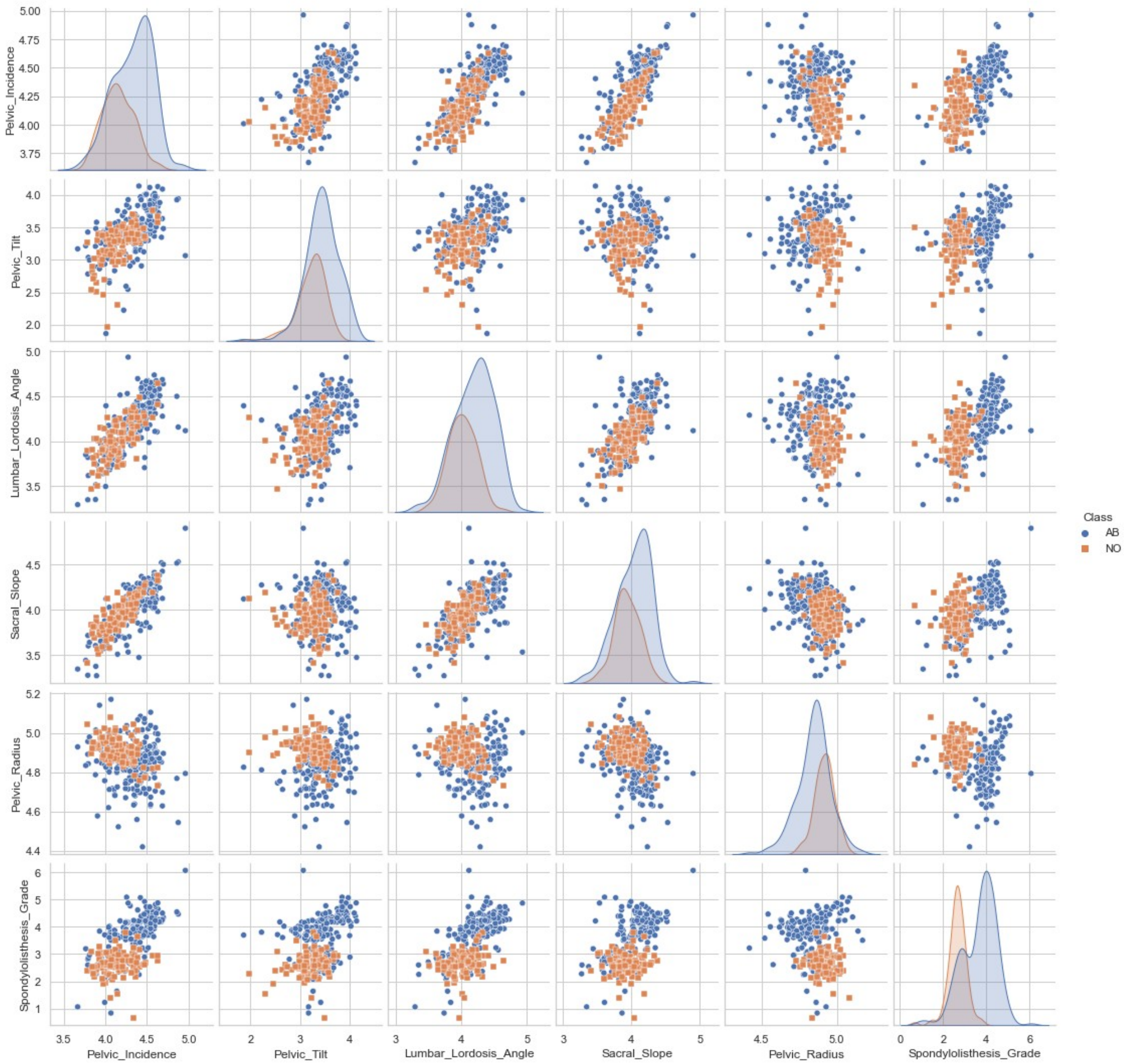

```
#Pairplot
sns.set(style="whitegrid")
sns.pairplot(df, hue="Class", diag_kind="kde", markers=["o", "s"])
plt.savefig('pairplot1.jpg')
plt.show()
```




```
df

     Pelvic_Incidence  Pelvic_Tilt  Lumbar_Lordosis_Angle
Sacral_Slope  \
0               63.03        22.55                  39.61
40.48
1               39.06        10.06                  25.02
29.00
2               68.83        22.22                  50.09
46.61
3               69.30        24.65                  44.31
44.64
4               49.71         9.65                  28.32
40.06
```

```
..                   ...          ...                       ...             ..
.
305              47.90        13.62                       36.00
34.29
306              53.94        20.72                       29.22
33.22
307              61.45        22.69                       46.17
38.75
308              45.25         8.69                       41.58
36.56
309              33.84         5.07                       36.64
28.77

     Pelvic_Radius  Spondylolisthesis_Grade Class
0            98.67                    -0.25    AB
1           114.41                     4.56    AB
2           105.99                    -3.53    AB
3           101.87                    11.21    AB
4           108.17                     7.92    AB
..             ...                      ...   ...
305         117.45                    -4.25    NO
306         114.37                    -0.42    NO
307         125.67                    -2.71    NO
308         118.55                     0.21    NO
309         123.95                    -0.20    NO

[310 rows x 7 columns]

#Shifting the data by 12
scale_df=df[['Pelvic_Incidence','Pelvic_Tilt',
'Lumbar_Lordosis_Angle','Sacral_Slope','Pelvic_Radius','Spondylolisthe
sis_Grade']].copy()
scale_df=scale_df + abs(12)

scale_df.describe()

       Pelvic_Incidence  Pelvic_Tilt  Lumbar_Lordosis_Angle
Sacral_Slope  \
count        310.000000   310.000000             310.000000
310.000000
mean          72.496484    29.542903              63.930710
54.953871
std           17.236109    10.008140              18.553766
13.422748
min           38.150000     5.450000              26.000000
25.370000
25%           58.432500    22.667500              49.000000
45.347500
50%           70.690000    28.360000              61.565000
54.405000
```

```
75%              84.880000    34.120000                75.000000
64.692500
max             141.830000    61.430000               137.740000
133.430000

       Pelvic_Radius  Spondylolisthesis_Grade
count     310.000000               310.000000
mean      129.920548                38.296742
std        13.317629                37.558883
min        82.080000                 0.940000
25%       122.710000                13.600000
50%       130.265000                23.765000
75%       137.467500                53.285000
max       175.070000               430.540000

#log transformation
log_transformed_df = np.log1p(scale_df)

log_transformed_df=pd.concat([log_transformed_df, df['Class']],
axis=1)

#Pairplot after log transformation
sns.set(style="whitegrid")
sns.pairplot(log_transformed_df, hue="Class", diag_kind="kde",
markers=["o", "s"])
plt.savefig('pairplot2.jpg')
plt.show()
```



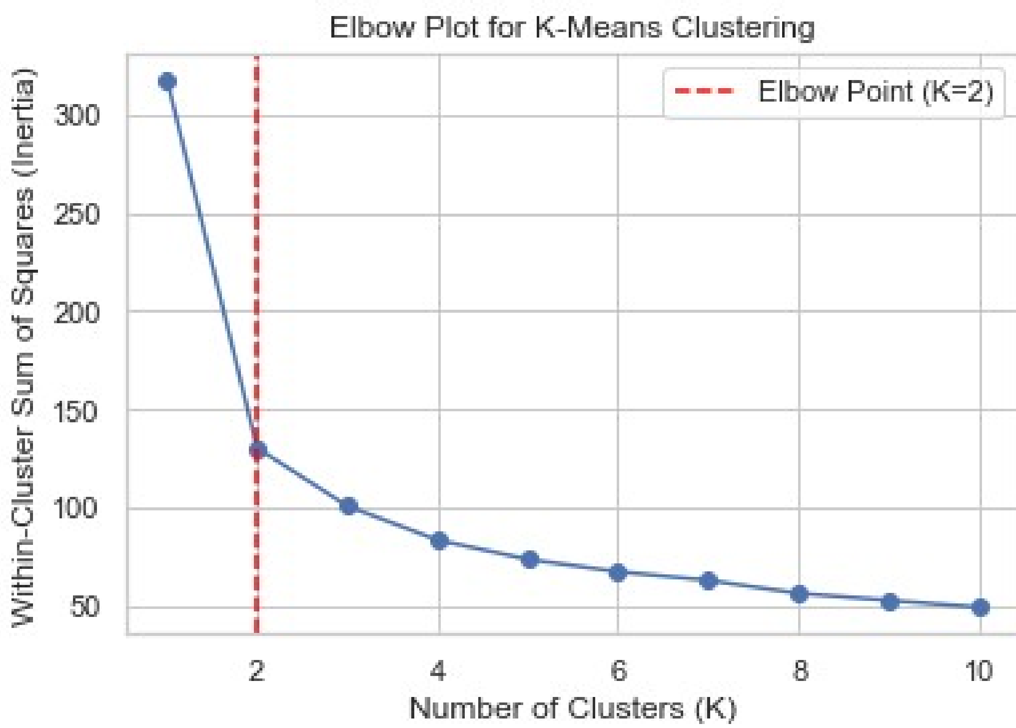

```
log_transformed_df

     Pelvic_Incidence  Pelvic_Tilt  Lumbar_Lordosis_Angle
Sacral_Slope  \
0            4.331128     3.570940               3.962906
3.979308
1            3.952397     3.138100               3.638112
3.737670
2            4.404644     3.561614               4.144562
4.087823
3            4.410371     3.628333               4.048475
4.054217
4            4.138521     3.120160               3.721347
3.971423
```

```
..                   ...           ...                        ...              ..
.
305          4.109233     3.281663                 3.891820
3.856299
306          4.203797     3.518091                 3.742894
3.833413
307          4.310128     3.574871                 4.080415
3.946424
308          4.064744     3.076851                 3.999668
3.903184
309          3.846738     2.894253                 3.904797
3.732178

     Pelvic_Radius  Spondylolisthesis_Grade Class
0         4.715548                 2.545531    AB
1         4.847410                 2.865624    AB
2         4.779039                 2.248129    AB
3         4.743801                 3.186766    AB
4         4.797195                 3.040706    AB
..             ...                      ...   ...
305       4.870990                 2.169054    NO
306       4.847096                 2.532108    NO
307       4.932097                 2.331173    NO
308       4.879387                 2.580974    NO
309       4.919616                 2.549445    NO

[310 rows x 7 columns]

#Label encoding the Class field
label_encoder = LabelEncoder()
log_transformed_df['Class'] =
label_encoder.fit_transform(log_transformed_df['Class'])
```

K-Means Clustering Unsupervised Model

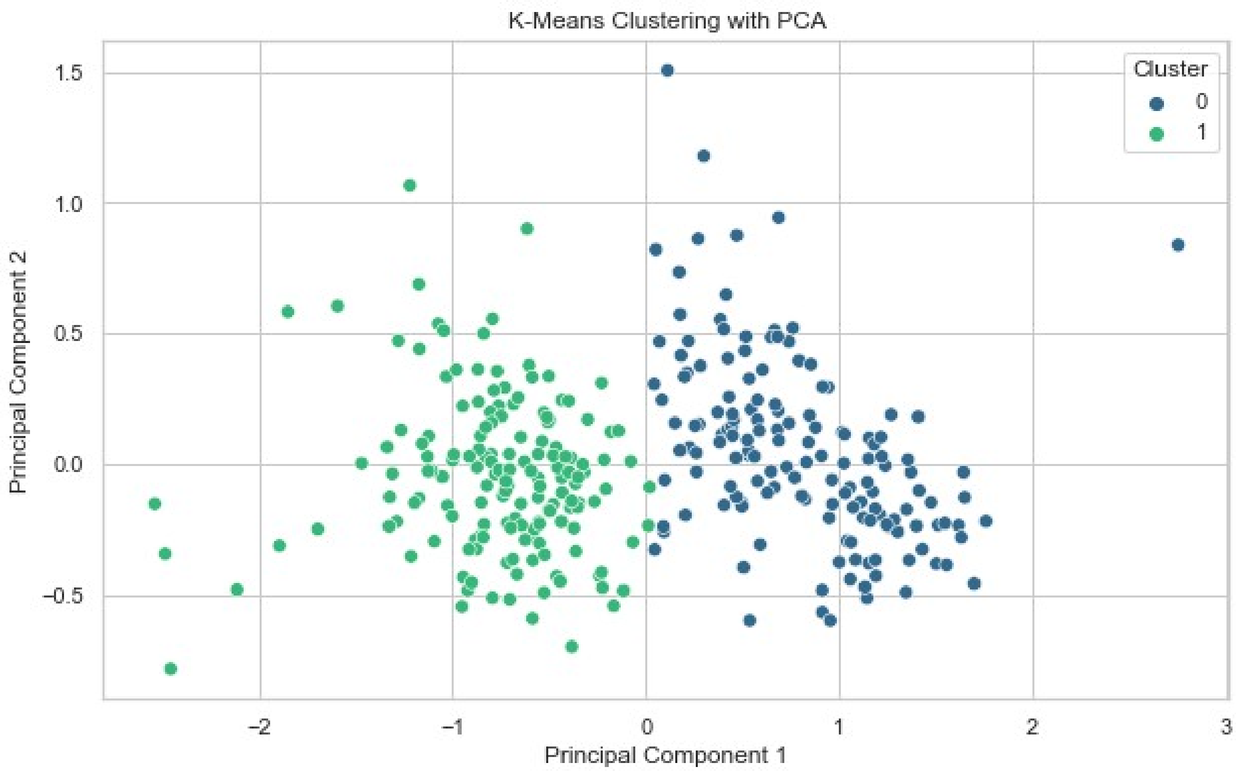
```
features =log_transformed_df[['Pelvic_Incidence','Pelvic_Tilt',
'Lumbar_Lordosis_Angle','Sacral_Slope','Pelvic_Radius','Spondylolisthe
sis_Grade']]

# Calculating inertia for different values of K
inertia_values = []
k_values = range(1, 11)

for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(features)
    inertia_values.append(kmeans.inertia_)
```

```
#Elbow plot
plt.plot(k_values, inertia_values, marker='o')
plt.title('Elbow Plot for K-Means Clustering')
plt.xlabel('Number of Clusters (K)')
plt.ylabel('Within-Cluster Sum of Squares (Inertia)')

plt.axvline(x=2, color='red', linestyle='--', label='Elbow Point
(K=2)')

plt.legend()
plt.show()
```




```
numerical_features= log_transformed_df[['Pelvic_Incidence',
'Pelvic_Tilt', 'Lumbar_Lordosis_Angle', 'Sacral_Slope',
'Pelvic_Radius', 'Spondylolisthesis_Grade']]

#Standard scaling the dataset
scaler = StandardScaler(with_std=False)
values = scaler.fit_transform(numerical_features)

#PCA
PCAthreshold = 2
pca = PCA(n_components=PCAthreshold, svd_solver="full")
reduced_data = pca.fit_transform(values)

# K-means clustering
kmeans = KMeans(n_clusters=2)
```

```
result_df = pd.DataFrame(data=reduced_data,
columns=[f'Component_{i+1}' for i in range(PCAthreshold)])
result_df['cluster'] = kmeans.fit_predict(reduced_data)

#ARI Score
ari_score = adjusted_rand_score(log_transformed_df['Class'],
result_df['cluster'])

print(f"Adjusted Rand Index (ARI): {ari_score}")

Adjusted Rand Index (ARI): 0.3352137850407571

#Scatter Plot after K-means
plt.figure(figsize=(10, 6))
sns.scatterplot(x=reduced_data[:, 0], y=reduced_data[:, 1],
hue=result_df['cluster'], palette='viridis', s=50)
plt.title('K-Means Clustering with PCA')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend(title='Cluster')
plt.show()
```



KNN Supervised Model

```
X = log_transformed_df[['Pelvic_Incidence','Pelvic_Tilt',
'Lumbar_Lordosis_Angle','Sacral_Slope','Pelvic_Radius','Spondylolisthe
sis_Grade']]
y = log_transformed_df['Class']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
k_values = range(1, 21)

# Cross Validation for each value of k
cv_scores = []

for k in k_values:
    knn = KNeighborsClassifier(n_neighbors=k)
    scores = cross_val_score(knn, X_train, y_train, cv=5,
scoring='accuracy')
    cv_scores.append(scores.mean())

# Finding the optimal k value
optimal_k = k_values[cv_scores.index(max(cv_scores))]

print(f"The optimal k value is: {optimal_k}")
```

```
The optimal k value is: 9
```

```
#KNN Classifier
knn_classifier = KNeighborsClassifier(n_neighbors=optimal_k)
knn_classifier.fit(X_train, y_train)

# Prediction
y_pred = knn_classifier.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)

print(f"Accuracy on the test set with optimal k: {accuracy:.2f}")
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"F1 Score: {f1:.2f}")
```

```
Accuracy on the test set with optimal k: 0.81
Precision: 0.67
Recall: 0.67
F1 Score: 0.67
```