DATA70132 COURSEWORK

SUPERVISED & UNSUPERVISED CLASSIFICATION

ID:11349153

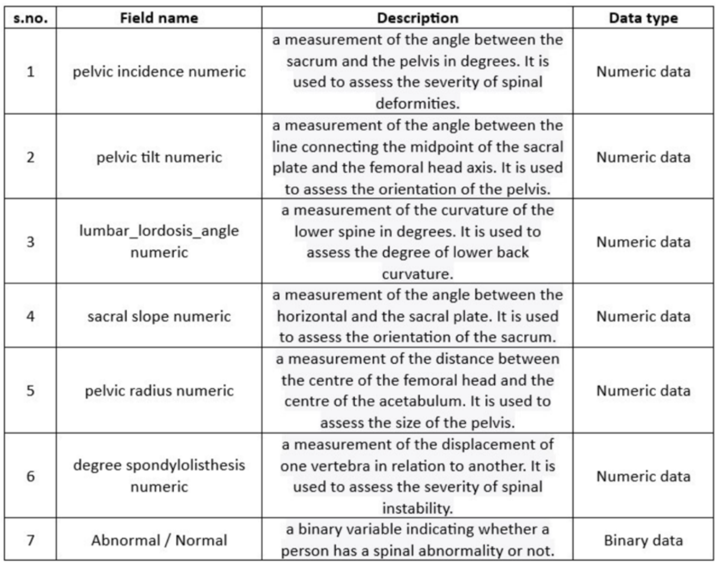
INTRODUCTION

Machine learning, a prevalent form of AI, employs statistical methods to train models using data. The integration of AI in healthcare holds promise for revolutionizing various aspects of the industry.

Here we will be working with one such biomedical dataset built by Dr. Henrique da Mota during a medical residence period in the Group of Applied Research in Orthopaedics (GARO) of the Centre MÃ©dico-Chirurgical de RÃ©adaptation des Massues, Lyon, France. Our main task here would be to classify the patient category based on their features using both supervised and unsupervised algorithms.

DATA

The data used here is derived from UCI Machine Learning Repository. A detailed representation is shown below.

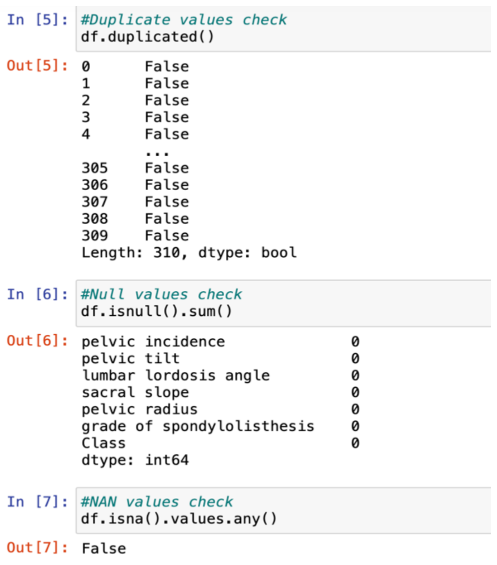


s.no.	Field name	Description	Data type
1	pelvic incidence numeric	a measurement of the angle between the sacrum and the pelvis in degrees. It is used to assess the severity of spinal deformities.	Numeric data
2	pelvic tilt numeric	a measurement of the angle between the line connecting the midpoint of the sacral plate and the femoral head axis. It is used to assess the orientation of the pelvis.	Numeric data
3	lumbar_lordosis_angle numeric	a measurement of the curvature of the lower spine in degrees. It is used to assess the degree of lower back curvature.	Numeric data
4	sacral slope numeric	a measurement of the angle between the horizontal and the sacral plate. It is used to assess the orientation of the sacrum.	Numeric data
5	pelvic radius numeric	a measurement of the distance between the centre of the femoral head and the centre of the acetabulum. It is used to assess the size of the pelvis.	Numeric data
6	degree spondylolisthesis numeric	a measurement of the displacement of one vertebra in relation to another. It is used to assess the severity of spinal instability.	Numeric data
7	Abnormal / Normal	a binary variable indicating whether a person has a spinal abnormality or not.	Binary data

Figure 1. Metadata of Vertebral Column dataset(https://www.linkedin.com/pulse/vertebral-column-dataset-muskan-gupta/)

The first thing that we need to check after importing the data is to see if its clean or not. By clean, we mean removing outliers, duplicate values, missing values, and NAN values if any.

The figure below confirms that there are no duplicate/missing/NAN values in the dataset.



```
In [5]: #Duplicate values check
        df.duplicated()

Out[5]: 0      False
        1      False
        2      False
        3      False
        4      False
               ...
        305    False
        306    False
        307    False
        308    False
        309    False
        Length: 310, dtype: bool

In [6]: #Null values check
        df.isnull().sum()

Out[6]: 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 [7]: #NAN values check
        df.isna().values.any()

Out[7]: False
```

Figure 3. Data cleaning

EXPLORATORY DATA ANALYSIS

We will start by checking for outliers in the dataset using boxplot which uses 5-point summary method to detect outliers.

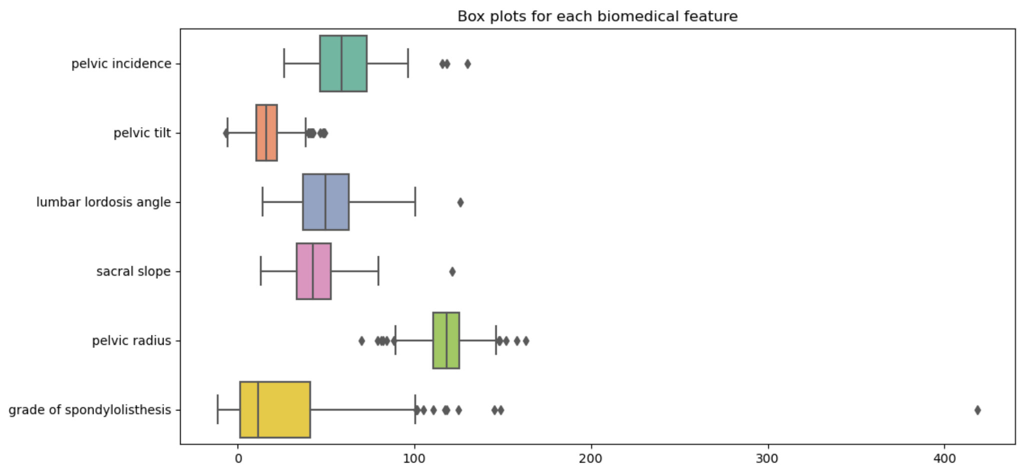


Figure 4. Outlier Detection

Outliers are observed in various features, requiring treatment before applying algorithms. In the case of this medical dataset, the strategy is to use imputation for handling outliers instead of outright removal. This approach is chosen due to the dataset's medical nature, where extreme values hold crucial information.

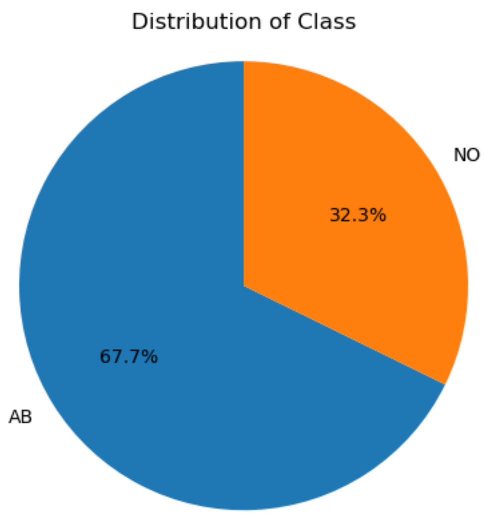


Figure 5

Visualizing the data, we can confirm its an imbalanced dataset having more people in abnormal category than normal.

SUMMARY STATISTICS

	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

Now, let’s look at the pair plot below to understand the relationships between the features.

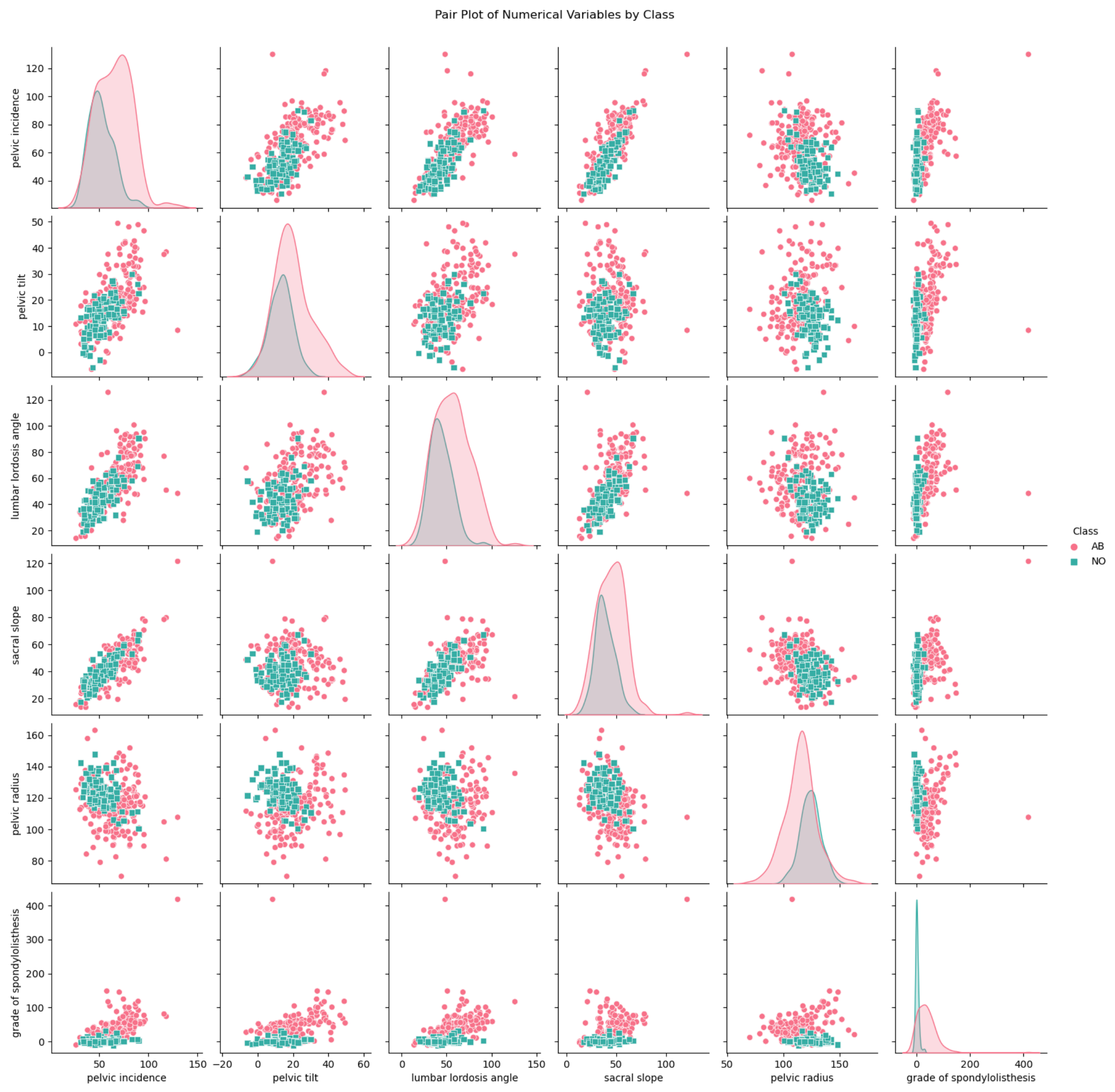


Figure 6.Pair plot

The figure illustrates notable linear relationships among variables, with pelvic incidence and sacral scope exhibiting strong correlation, whereas pelvic radius and pelvic incidence do not align as closely. To gain a clearer understanding, a correlation matrix is employed. Additionally, it is observed that one variable's data is right-skewed, prompting the need for data scaling to ensure consistency before applying machine learning algorithms. The standard scaler function from scikit-learn is utilized for this standardization process..

Now, we will be checking the feature importance to identify which features have the most impact on determining class labels using feature importance function from random forest classifier model.

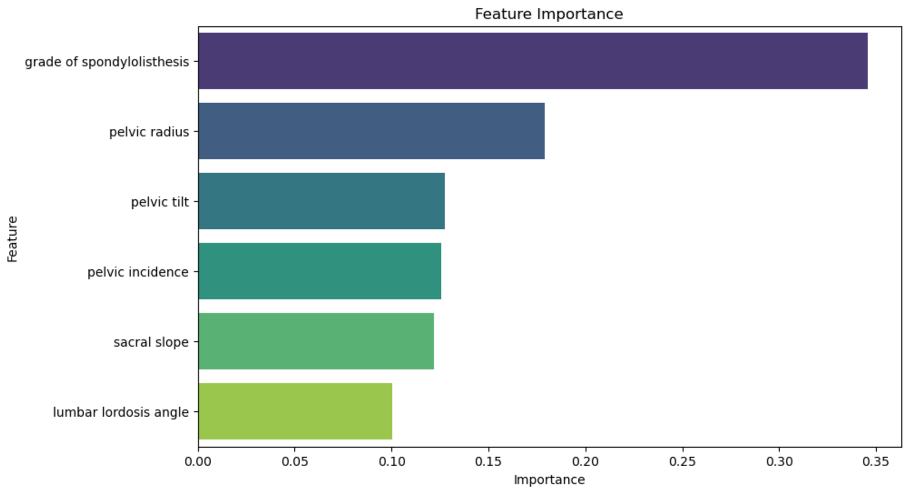


Figure 8. Feature Importance of independent features

So, mainly 2 features have the most impact on class labels so we can use dimensionality reduction techniques to change the dataset to 2D space. The main use of PCA components is that it records the maximal variance among all the features so we can focus on only those components for model training.

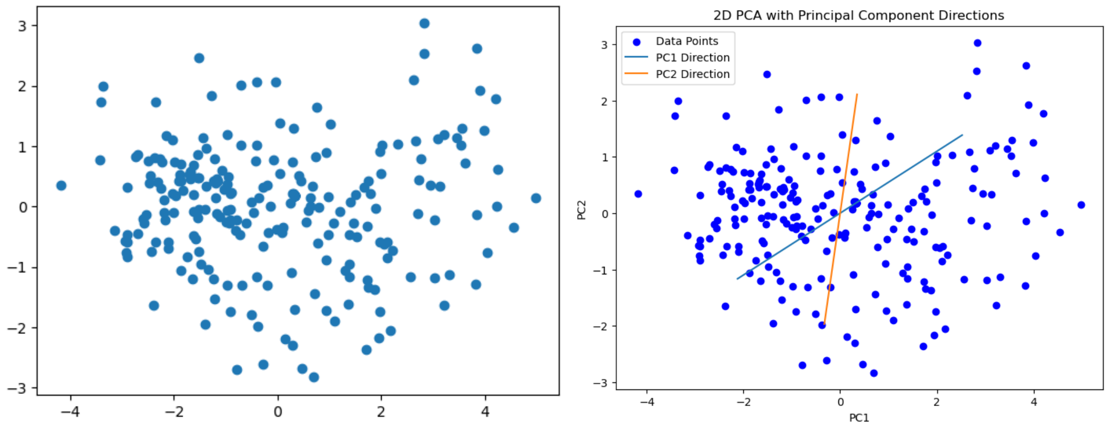
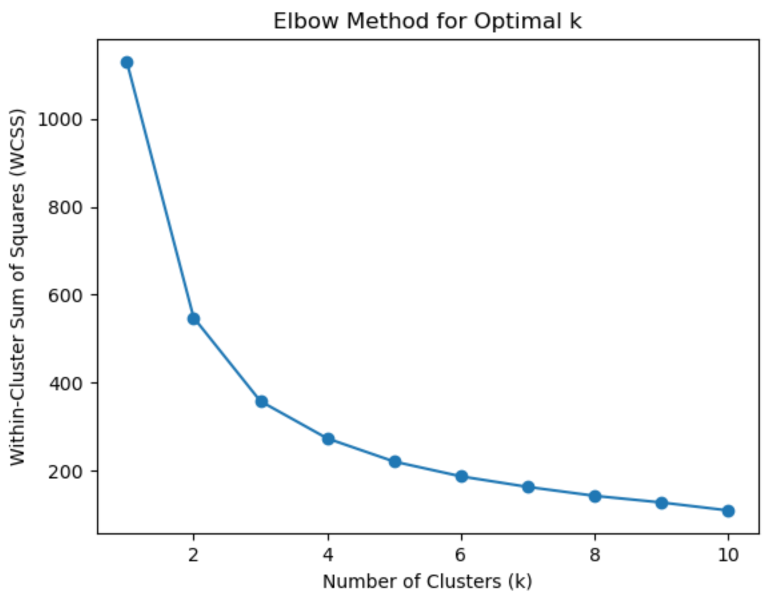


Figure 9. Scatterplot before and after PCA

After applying PCA to the dataset, the next step involves applying classification algorithms. However, for k-means clustering, determining the optimal value of k is crucial. The Within-Cluster-Sum-of-Squares (WCSS) method is employed for this purpose, measuring the sum of squared distances within clusters and visualizing the results on a graph.



The elbow curve graph, named for its shape, helps identify the optimal value of k where there is no significant change in WCSS value, resulting in a smoother curve. Hence chose 3.

MODELLING

SUPERVISED

Supervised classification, an approach where algorithms learn from labelled training data for making predictions, employs Support Vector Machine (SVM). SVM excels in handling complex decision boundaries and high-dimensional data. It seeks the hyperplane maximizing the distance between class margins to distinguish data points distinctly, preventing overlap. Leveraging support vectors and marginal planes, SVM proves robust and adaptable to various datasets, suitable for both linear and non-linear classification tasks. The SVM objective function is represented as:

$f(x)=\text{sign}\,(\mathbf{w}\cdot\mathbf{x}+b)$

where w is the weight vector, x is the input feature vector, b is the bias term, and {sign} returns the sign of the expression.

UNSUPERVISED

Clustering, an unsupervised learning technique for classification tasks, plays a vital role in pattern recognition, data exploration, and segmentation. K-Means clustering is chosen for its simplicity and efficiency. This method iteratively assigns data points to the nearest centroid, creating k clusters based on similarity.

The K-Means algorithm involves two main steps: assignment and update. In the assignment step, distances between each data point and cluster centroids are calculated, assigning points to the nearest cluster. The update step then recalculates centroids based on the mean of data points in each cluster. This process repeats until convergence.

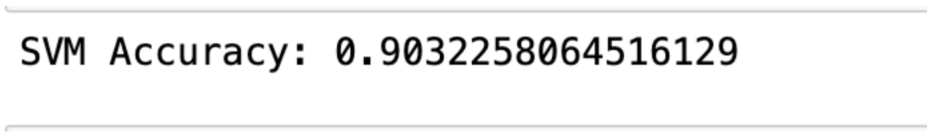
The goal in K-Means clustering is to minimize Within-Cluster Sum of Squares (WCSS), where smaller values indicate tighter and more cohesive clusters. It is represented by the equation:

$$\mathrm{WCSS} = \sum_{i=1}^{k} \sum_{j=1}^{n_i} |x_{ij} - c_i|^2$$

where:
k is the number of clusters.
ni is the number of data points in cluster *i*.
xij is the *j*-th data point in cluster *i*.
ci is the centroid of cluster *i*.

RESULTS

SVM



```
SVM Accuracy: 0.9032258064516129
```

This signifies that our model works perfectly on test data with 90% accuracy.

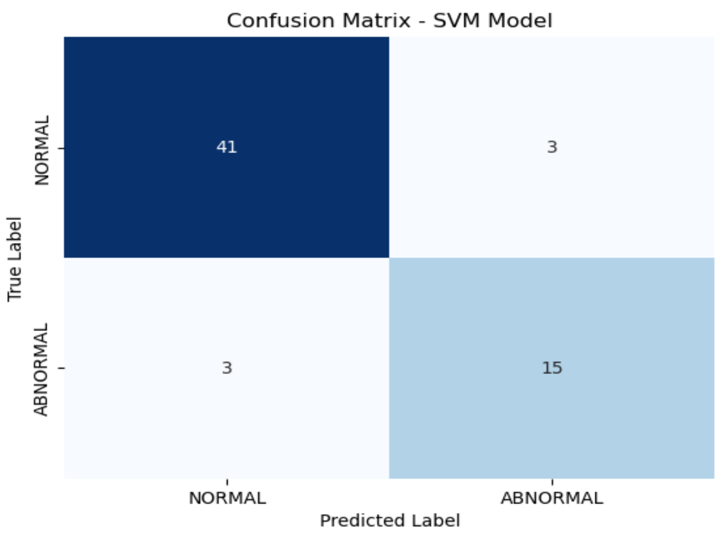


Figure 10.Confusion Matrix

From confusion matrix, we can clearly understand the model performance, which shows that it was able to predict 41 normal and 15 abnormal categories correctly while it failed in 6 cases.

Evaluation Metrics

Recall
83.3%

Specificity
93.2%

Precision
83.3%

F1 Score
83.3%

Above metrics prove that this model works decently. Let's look at the illustrations below for better understanding.

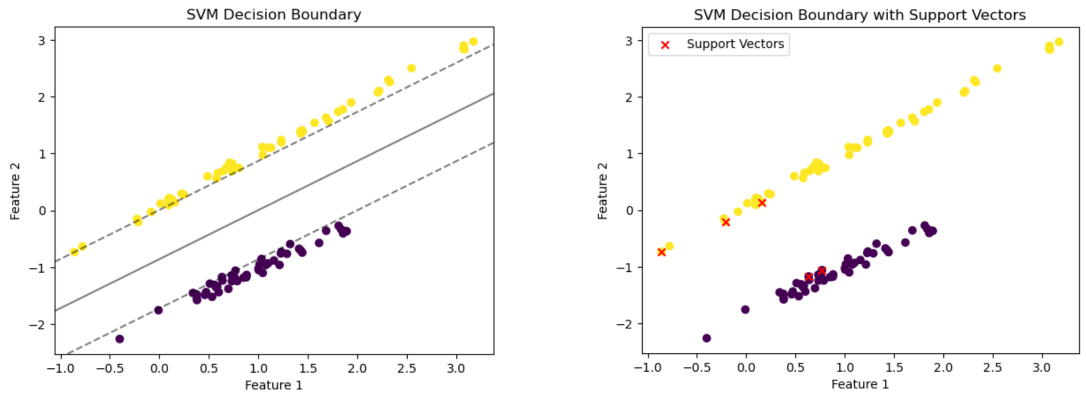


Figure 11.SVM Working Principle

Fig. 11 shows the support vectors involved in deciding margins and was also able to partition the data points accurately without overlapping.

K-Means

This algorithm also managed to identify inherent patterns within the data, in the form of clusters, without prior knowledge of the patient status.

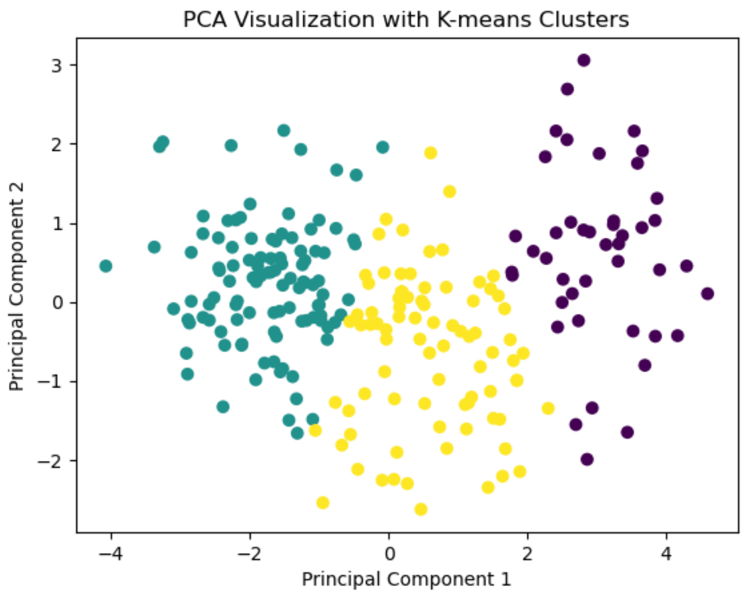


Figure 12

Above figure shows that there are 3 different clusters based on similar features indicating there might be another category other than normal and abnormal, probably disk hernia.

CONCLUSION

Comparing both the models, we can say that both the methodologies contribute to a comprehensive analysis of the dataset. While one method was able to detect the underlying patterns well, the other was able to identify the main factors influencing result.

To be precise, we can use K-means clustering for identifying the clusters based on similarities and this in turn would help us decide the features in supervised learning leading to more accurate predictions. On the other hand, SVM is a predictive model based on labelled data, so we can make use of those results to validate the clusters we found on unsupervised method.

REFERENCES

- Cortes, C., & Vapnik, V. (1995). 20(3), 273-297
- MacQueen, J. (1967). (Vol. 1, No. 14, pp. 281-297).

CODE

In [1]:
```
# Importing necessary libraries
from sklearn.cluster import KMeans
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.decomposition import PCA
from sklearn.datasets import make_classification
from sklearn.svm import SVC
from sklearn.model_selection import RandomizedSearchCV
from sklearn.metrics import confusion_matrix



columns = [ 'pelvic incidence', 'pelvic tilt', 'lumbar lordosis angle', 'sa
'pelvic radius','grade of spondylolisthesis', 'Class']

df = pd.read_csv(r"/Users/abhinandandas/Downloads/vertebral.dat", header =
```

In [2]:
```
# To display first 5 records
df.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

In [3]:
```
#Summary statistics
df.describe()
```

Out[3]:

	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 [4]:
```
#Checking duplicate values
df.duplicated()
```

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

In [6]:
```
#Checking null values
df.isnull().sum()
```

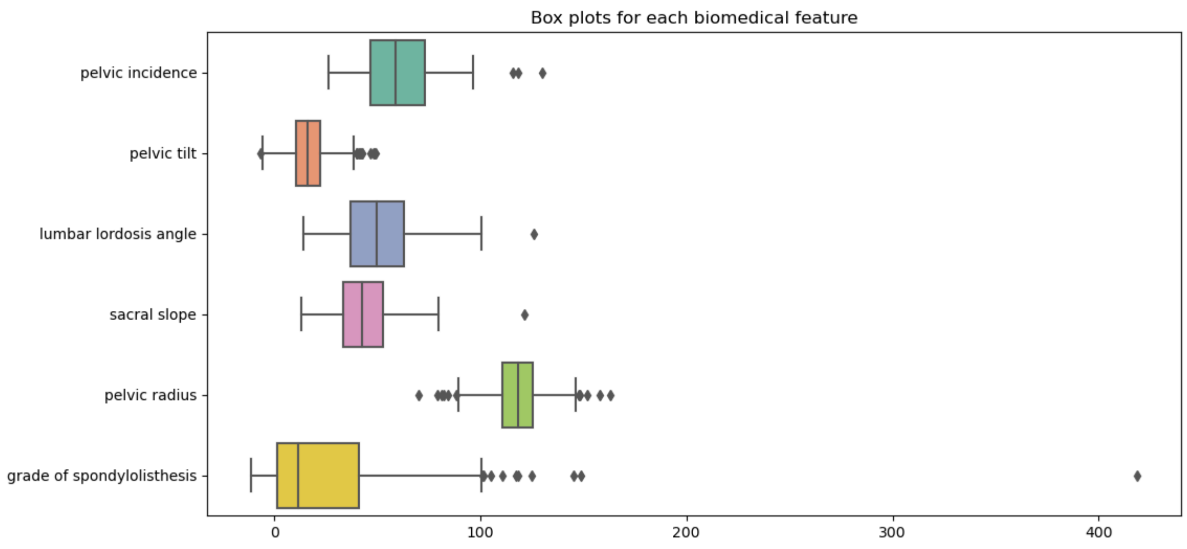
Out[6]:
```
pelvic incidence            0
pelvic tilt                 0
lumbar lordosis angle       0
sacral slope                0
pelvic radius               0
grade of spondylolisthesis  0
Class                       0
dtype: int64
```

In [7]:
```
#Checking NAN values
df.isna().values.any()
```

Out[7]:
```
False
```

In [8]:
```
X = df.drop('Class', axis=1)  # Features
y = df['Class']

# Data splitting
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra
```

In [9]:
```
#Boxplot for outlier detection

plt.figure(figsize=(12, 6))
sns.boxplot(data=df, orient="h", palette="Set2")
plt.title('Box plots for each biomedical feature')
plt.show()
```

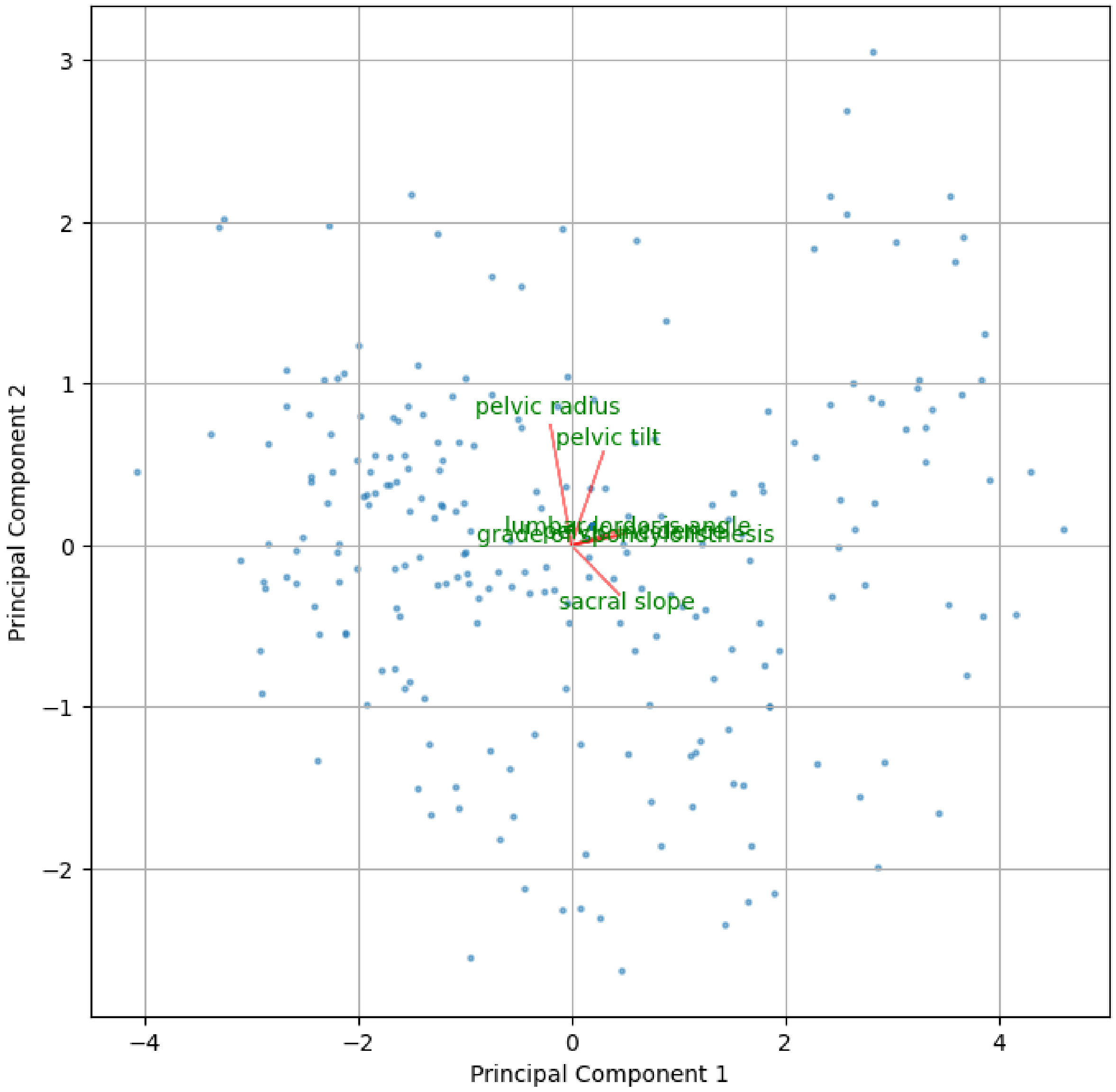


In [10]:
```
# Calculating Mean Absolute Deviation (MAD)
mad_values = X_train.apply(lambda x: np.abs(x - x.median()).median() / 0.67


# Threshold for identifying outliers
threshold_mad = 3

outliers_mad = (np.abs(X_train - X_train.mean()) > threshold_mad * mad_value

# Imputing outliers
X_train_imputed = X_train.copy()
X_train_imputed[outliers_mad] = X_train.mean()
```

In [11]:
```
#Standardization
scaler = StandardScaler()
X_train_std = scaler.fit_transform(X_train_imputed)
X_test_std = scaler.transform(X_test)
```

In [12]:
```
# Applying PCA for dimensionality reduction

pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_train_std)
```

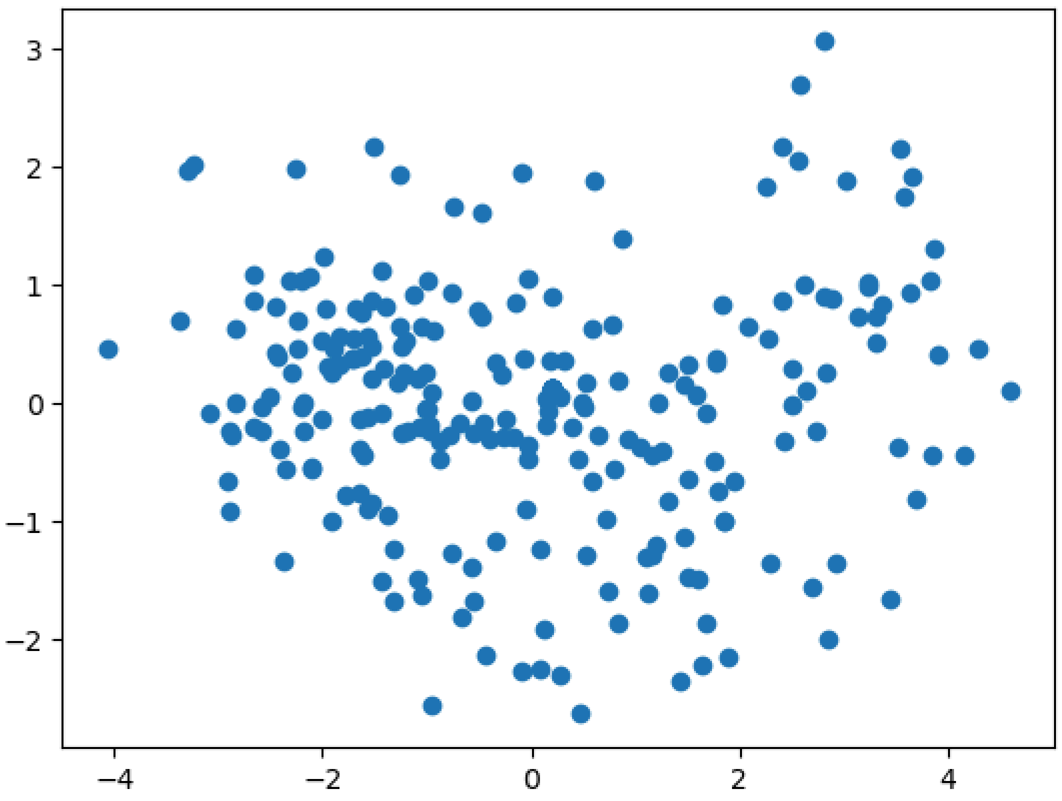
In [16]:
```
#BIPLOT to show variance percentage of principal components
feature_labels = ['pelvic incidence', 'pelvic tilt', 'lumbar lordosis angle
'pelvic radius','grade of spondylolisthesis']

# Replace with your feature names
biplot(X_pca[:, :2], np.transpose(pca.components_[:2, :]), labels=feature_la
plt.show()
```



In [17]:
```
#Before applying kmeans

plt.scatter(X_pca[:, 0], X_pca[:, 1],cmap='viridis')
```

```
/var/folders/5x/md3ppjv92893qyyrvrp7tsvr0000gn/T/ipykernel_6696/783658947.p
y:3: UserWarning: No data for colormapping provided via 'c'. Parameters 'cm
ap' will be ignored
  plt.scatter(X_pca[:, 0], X_pca[:, 1],cmap='viridis')
```

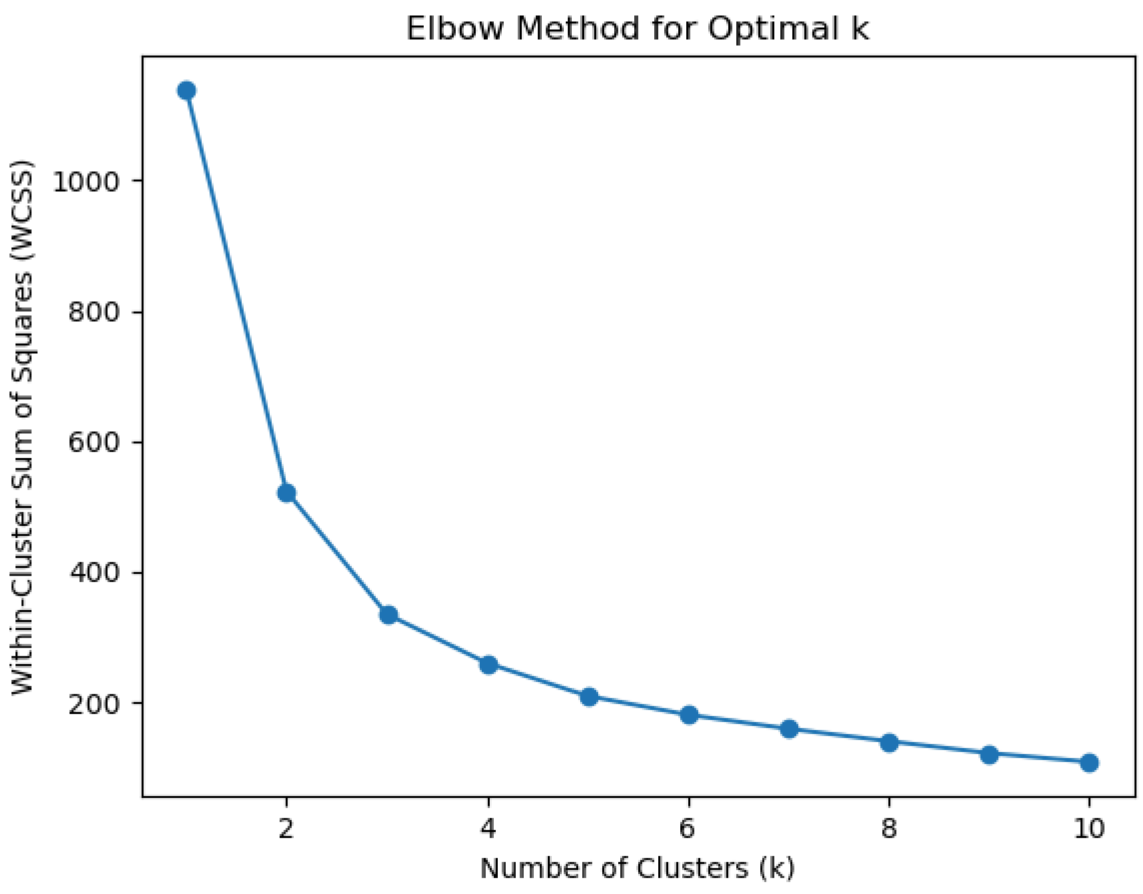
Out[17]:
```
<matplotlib.collections.PathCollection at 0x161767550>
```



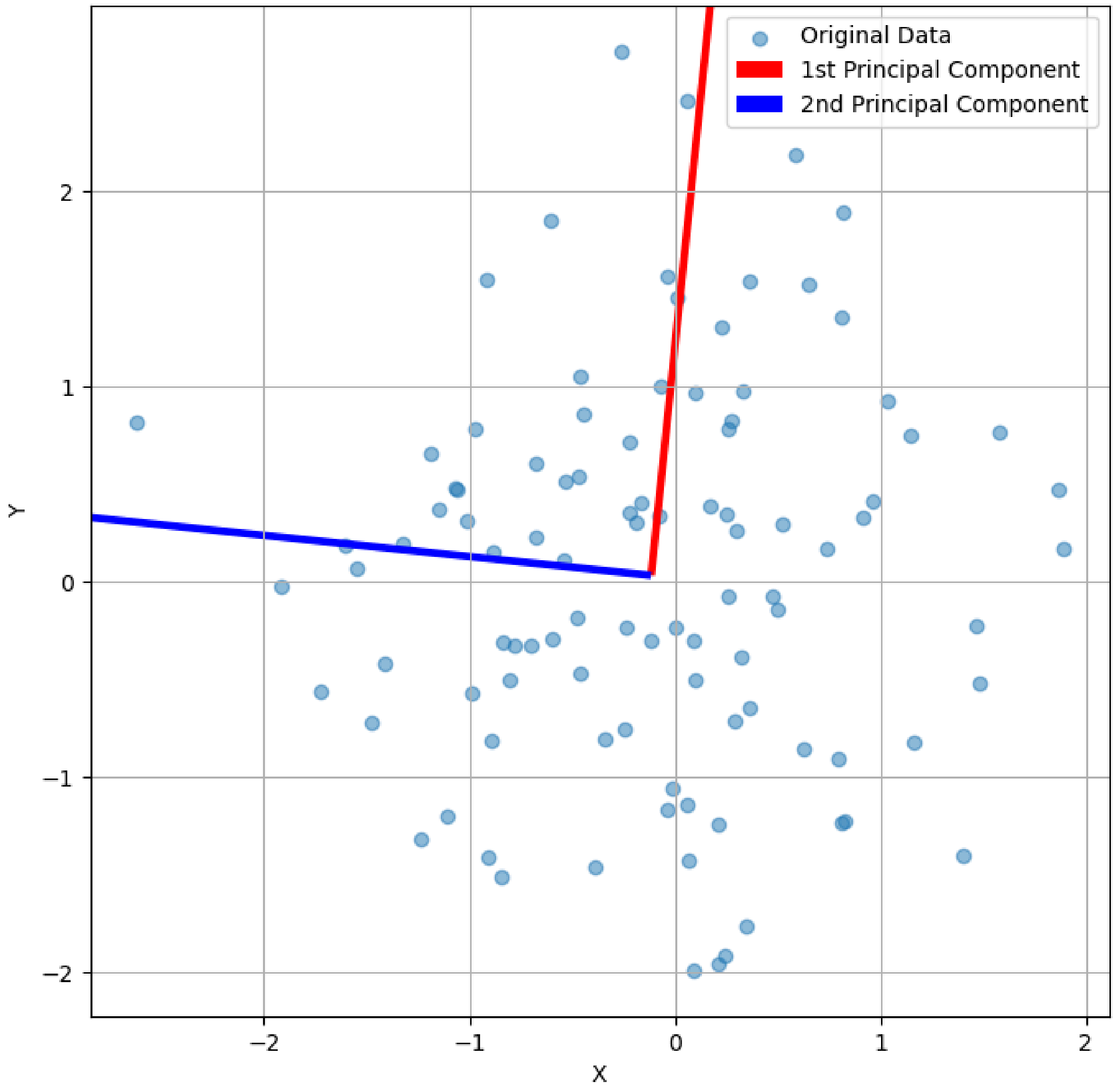
In [18]:

```
#Computing k value for clusters

import matplotlib.pyplot as plt

# list to store WCSS values for different k
wcss = []

# range of values to try
k_values = range(1, 11)

# Calculate WCSS for each k
for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(X_pca)
    wcss.append(kmeans.inertia_)


plt.plot(k_values, wcss, marker='o')
plt.title('Elbow Method for Optimal k')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Within-Cluster Sum of Squares (WCSS)')
plt.show()
```

```
/Users/abhinandandas/anaconda3/lib/python3.11/site-packages/sklearn/cluste
r/_kmeans.py:1412: FutureWarning: The default value of `n_init` will change
from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress
the warning
  super()._check_params_vs_input(X, default_n_init=10)
/Users/abhinandandas/anaconda3/lib/python3.11/site-packages/sklearn/cluste
r/_kmeans.py:1412: FutureWarning: The default value of `n_init` will change
from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress
the warning
  super()._check_params_vs_input(X, default_n_init=10)
/Users/abhinandandas/anaconda3/lib/python3.11/site-packages/sklearn/cluste
r/_kmeans.py:1412: FutureWarning: The default value of `n_init` will change
from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress
the warning
  super()._check_params_vs_input(X, default_n_init=10)
/Users/abhinandandas/anaconda3/lib/python3.11/site-packages/sklearn/cluste
r/_kmeans.py:1412: FutureWarning: The default value of `n_init` will change
from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress
the warning
  super()._check_params_vs_input(X, default_n_init=10)
/Users/abhinandandas/anaconda3/lib/python3.11/site-packages/sklearn/cluste
r/_kmeans.py:1412: FutureWarning: The default value of `n_init` will change
from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress
the warning
  super()._check_params_vs_input(X, default_n_init=10)
/Users/abhinandandas/anaconda3/lib/python3.11/site-packages/sklearn/cluste
r/_kmeans.py:1412: FutureWarning: The default value of `n_init` will change
from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress
the warning
  super()._check_params_vs_input(X, default_n_init=10)
/Users/abhinandandas/anaconda3/lib/python3.11/site-packages/sklearn/cluste
r/_kmeans.py:1412: FutureWarning: The default value of `n_init` will change
from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress
the warning
  super()._check_params_vs_input(X, default_n_init=10)
/Users/abhinandandas/anaconda3/lib/python3.11/site-packages/sklearn/cluste
r/_kmeans.py:1412: FutureWarning: The default value of `n_init` will change
from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress
the warning
  super()._check_params_vs_input(X, default_n_init=10)
/Users/abhinandandas/anaconda3/lib/python3.11/site-packages/sklearn/cluste
r/_kmeans.py:1412: FutureWarning: The default value of `n_init` will change
from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress
the warning
  super()._check_params_vs_input(X, default_n_init=10)
/Users/abhinandandas/anaconda3/lib/python3.11/site-packages/sklearn/cluste
r/_kmeans.py:1412: FutureWarning: The default value of `n_init` will change
from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress
the warning
  super()._check_params_vs_input(X, default_n_init=10)
```

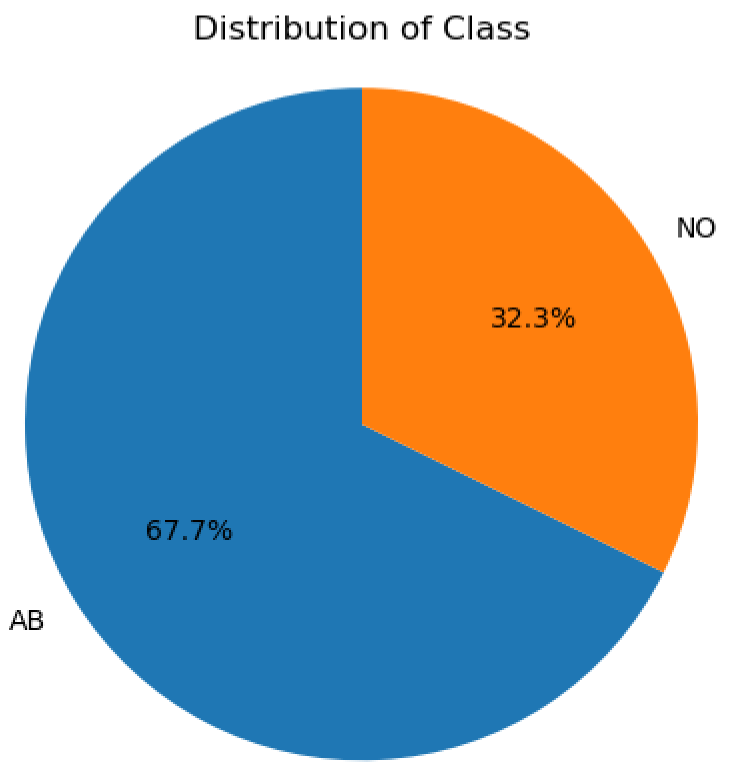


In [19]:
```
np.random.seed(42)
data = np.random.randn(100, 2)

# Fit PCA
pca = PCA(n_components=2)
pca.fit(data)
plt.figure(figsize=(8, 8))
plt.scatter(data[:, 0], data[:, 1], alpha=0.5, label='Original Data')

# Plotting PCA directions
origin = np.mean(data, axis=0)
scale = 2  # Adjust the scale for visualization
plt.quiver(*origin, *scale * pca.components_[0], color='red', scale=scale,
plt.quiver(*origin, *scale * pca.components_[1], color='blue', scale=scale,

plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
plt.grid(True)
plt.show()
```



In [20]:

```
class_counts = df['Class'].value_counts()
plt.pie(class_counts, labels=class_counts.index, autopct='%1.1f%%', startang
plt.axis('equal')  # Equal aspect ratio ensures the pie chart is circular.
plt.title('Distribution of Class')
plt.show()
```

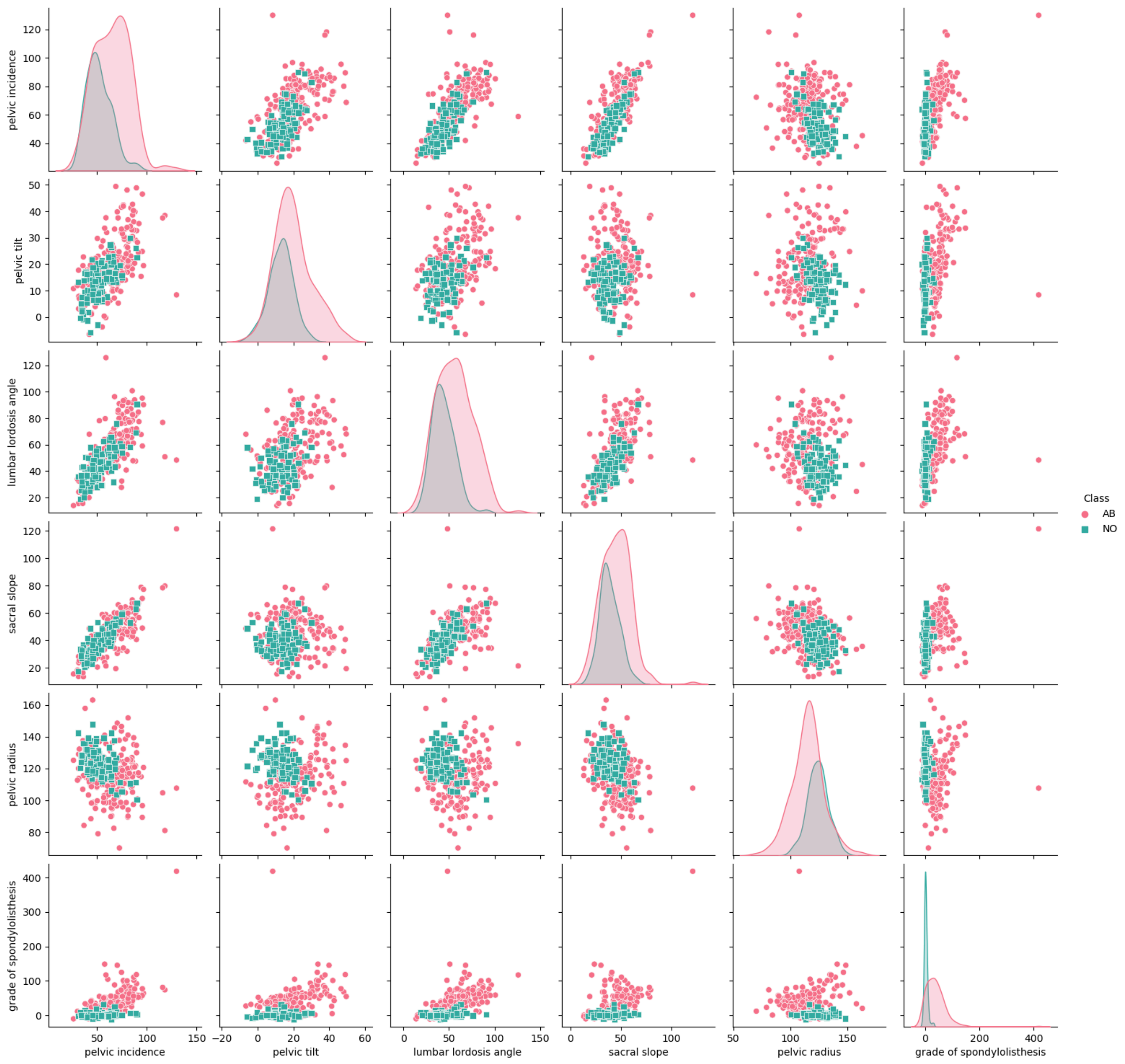


In [21]:

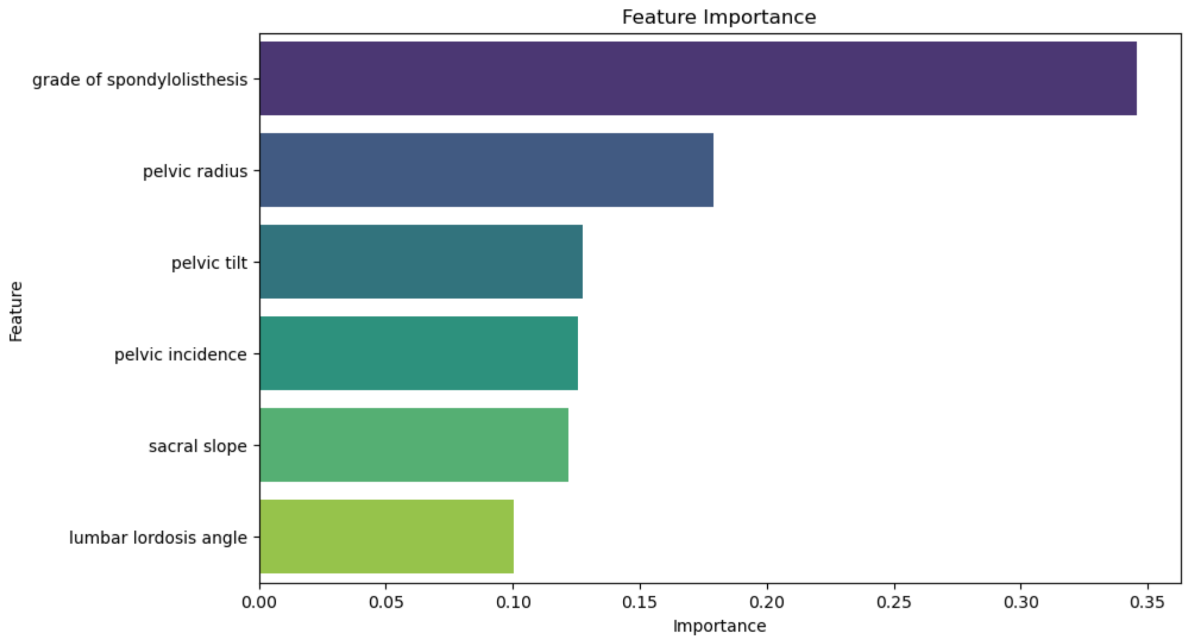
```
sns.pairplot(df, hue='Class', markers=['o', 's'], palette='husl')
plt.suptitle('Pair Plot of Numerical Variables by Class', y=1.02)
plt.show()
```

```
/Users/abhinandandas/anaconda3/lib/python3.11/site-packages/seaborn/axisgri
d.py:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

Pair Plot of Numerical Variables by Class



In [23]:
```
#Feature Importance Calculation
from sklearn.ensemble import RandomForestClassifier
X = df.drop("Class", axis=1)  # Features
y = df["Class"]  # Target variable

# Initializing Random Forest Classifier
rf_classifier = RandomForestClassifier(random_state=42)

# Model fitting
rf_classifier.fit(X, y)

# feature importance values
feature_importances = rf_classifier.feature_importances_


feature_importance_df = pd.DataFrame({"Feature": X.columns, "Importance": fe

# Sorting feature values
feature_importance_df = feature_importance_df.sort_values(by="Importance", a

# Plotting feature importance
plt.figure(figsize=(10, 6))
sns.barplot(x="Importance", y="Feature", data=feature_importance_df, palette
plt.title("Feature Importance")
plt.show()
```



In [24]:
```
#Hyperparameter tuning

from sklearn.model_selection import RandomizedSearchCV

param_dist = {'C': [0.1, 1, 10, 100], 'kernel': ['linear', 'rbf'], 'gamma':
svm_model = SVC()
random_search = RandomizedSearchCV(svm_model, param_dist, n_iter=10, cv=5, s
random_search.fit(X_train_std, y_train)

best_params = random_search.best_params_
best_model = random_search.best_estimator_
print("Best Parameters:", best_params)
print("Best Cross-Validated Accuracy: {:.2f}".format(random_search.best_sco
print("Best Model:")
print(best_model)
```

```
Best Parameters: {'kernel': 'rbf', 'gamma': 0.1, 'C': 10}
Best Cross-Validated Accuracy: 0.86
Best Model:
SVC(C=10, gamma=0.1)
```

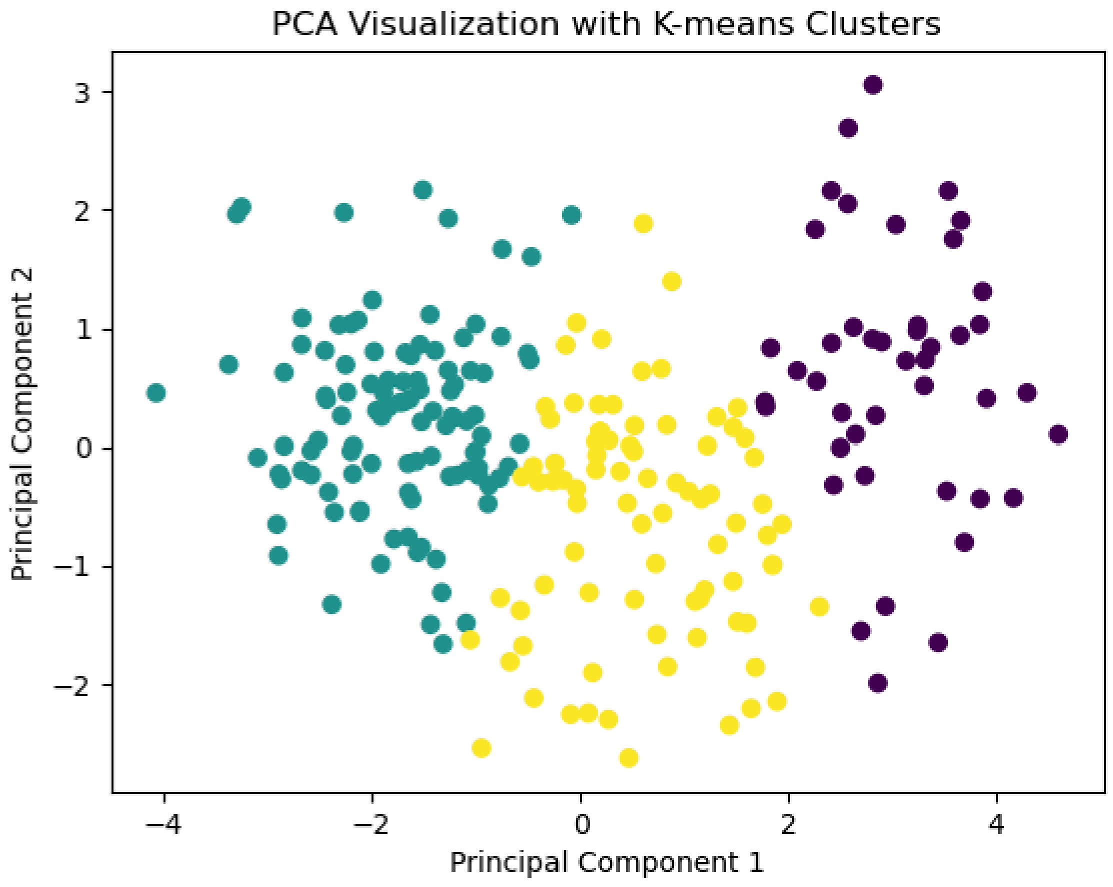
In [25]:
```
# Supervised Classification Model: SVM
svm_classifier = SVC(kernel='linear', C=1)
svm_classifier.fit(X_train_std, y_train)
y_pred = svm_classifier.predict(X_test_std)

# Accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"SVM Accuracy: {accuracy}")
```

```
SVM Accuracy: 0.9032258064516129
```

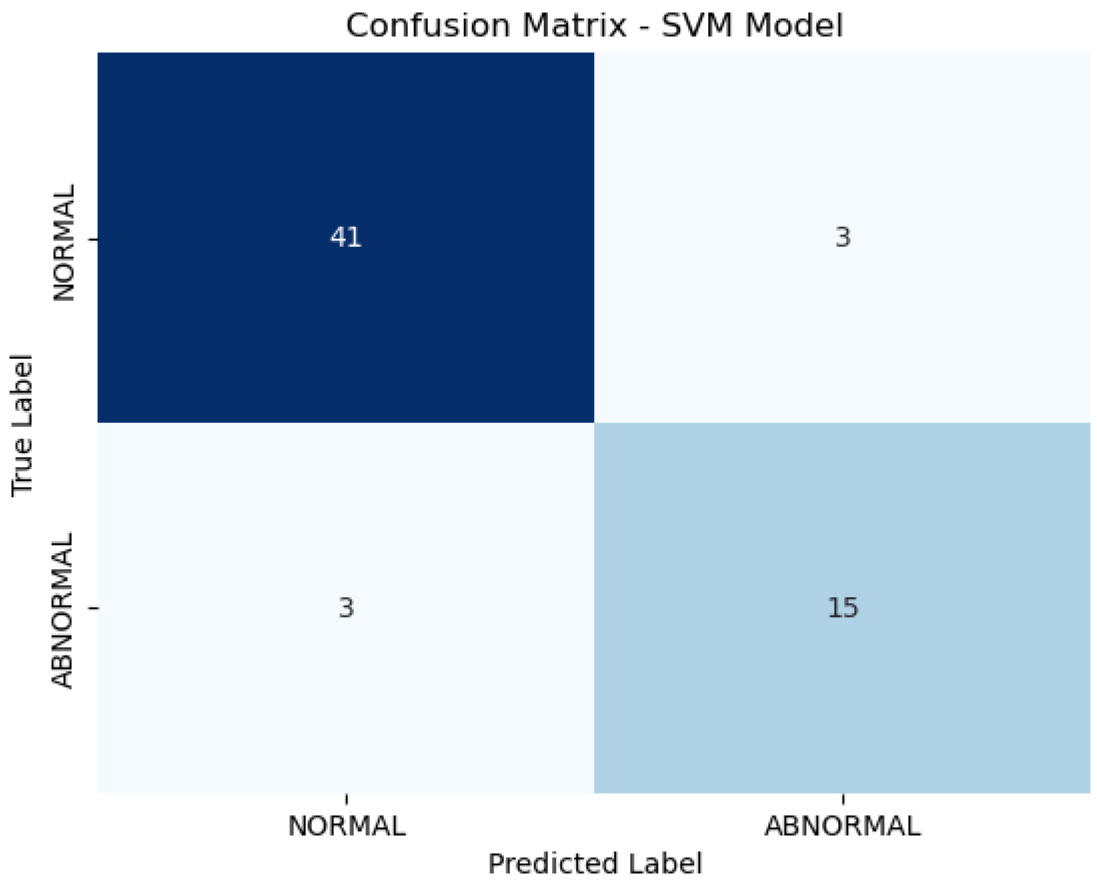
In [26]:
```
# Unsupervised Clustering: K-means
kmeans = KMeans(n_clusters=3, random_state=42)
clusters = kmeans.fit_predict(X_pca)
```

```
/Users/abhinandandas/anaconda3/lib/python3.11/site-packages/sklearn/cluste
r/_kmeans.py:1412: FutureWarning: The default value of `n_init` will change
from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress
the warning
  super()._check_params_vs_input(X, default_n_init=10)
```

In [27]:
```
# Scatter plot with clusters
plt.scatter(X_pca[:, 0], X_pca[:, 1], c= clusters, cmap='viridis')
plt.title('PCA Visualization with K-means Clusters')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.show()
```

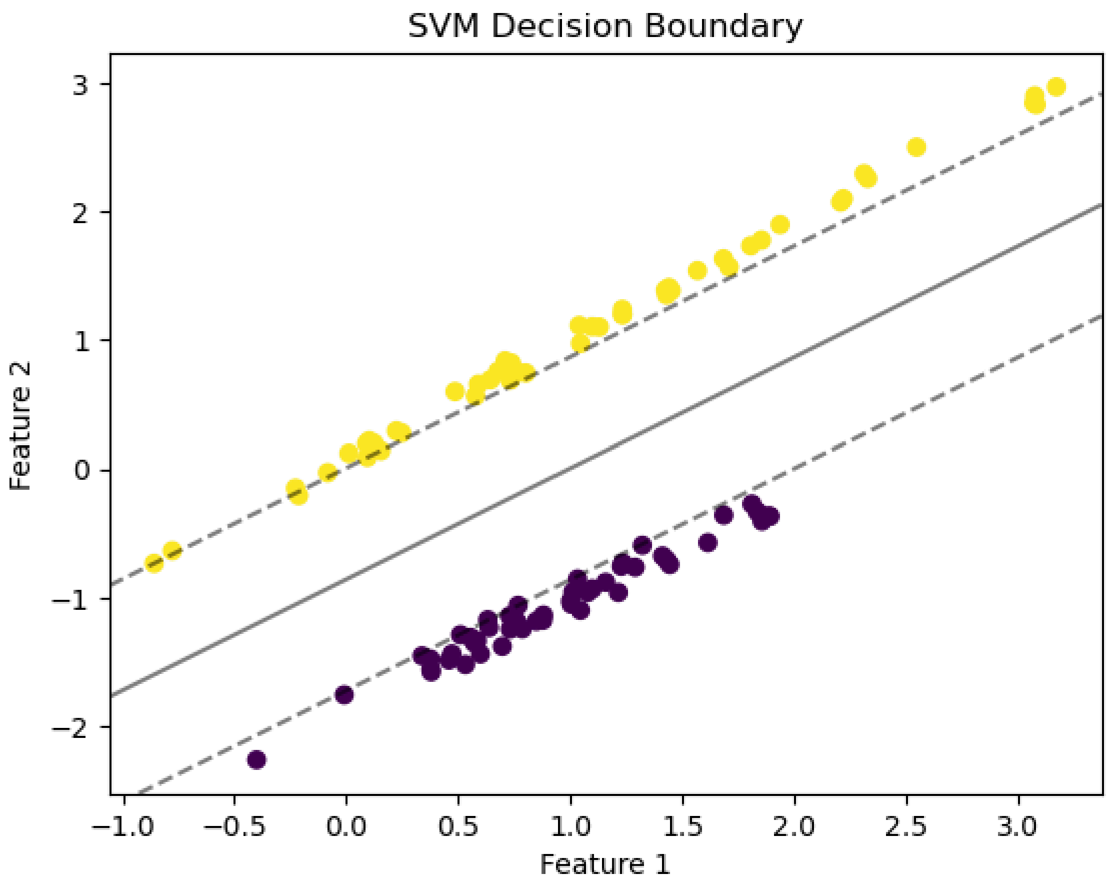


In [28]:
```
y_pred = svm_classifier.predict(X_test_std)

# Confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Plotting the matrix
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', cbar=False,
            xticklabels=['NORMAL', 'ABNORMAL'], yticklabels=['NORMAL', 'ABN
plt.title('Confusion Matrix - SVM Model')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```

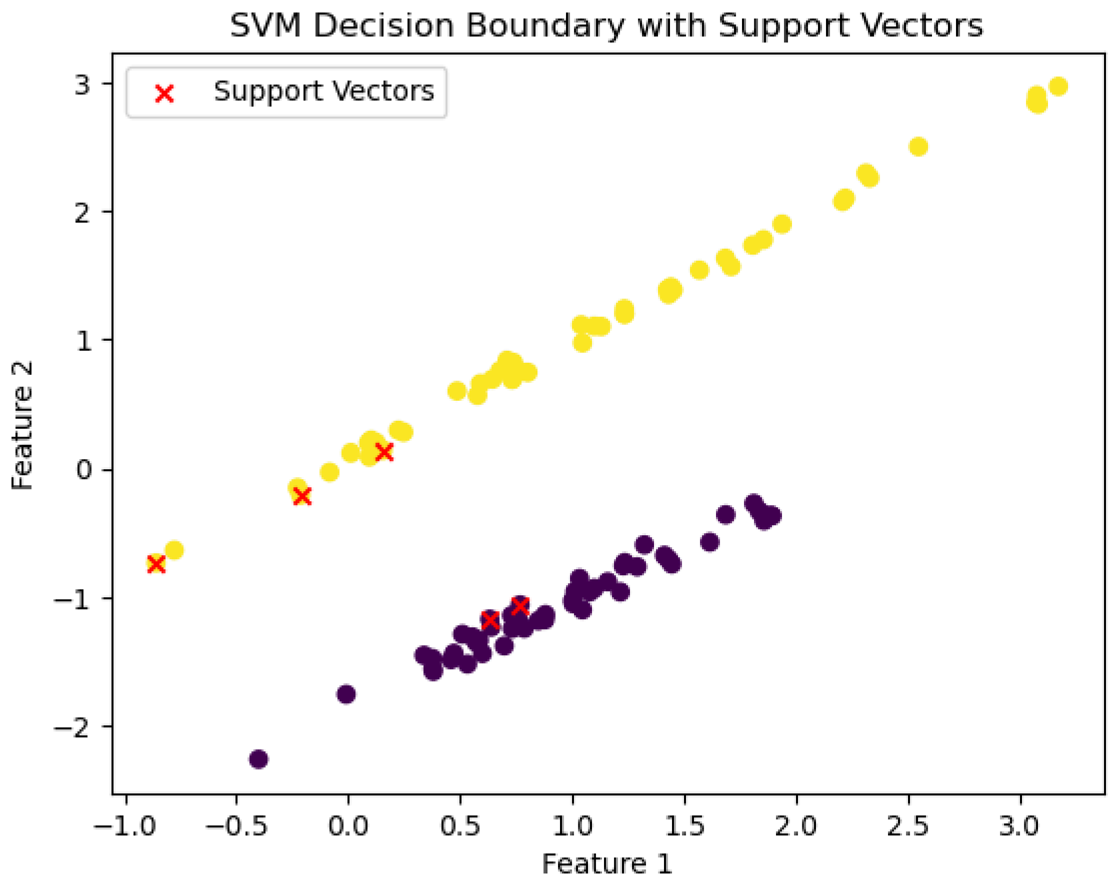


In [30]:
```
from sklearn.datasets import make_classification
from sklearn.svm import SVC
import matplotlib.pyplot as plt
import numpy as np


X, y = make_classification(
    n_samples=100,
    n_features=2,
    n_informative=2,
    n_redundant=0,
    n_classes=2,
    n_clusters_per_class=1,
    random_state=42
)

# Fitting SVM model
svm = SVC(kernel='linear')
svm.fit(X, y)

# Plotting decision boundary
plt.scatter(X[:, 0], X[:, 1], c=y, cmap='viridis')
plt.title('SVM Decision Boundary')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')


ax = plt.gca()
xlim = ax.get_xlim()
ylim = ax.get_ylim()

# grid for evaluating model
xx, yy = np.meshgrid(np.linspace(xlim[0], xlim[1], 100), np.linspace(ylim[0
Z = svm.decision_function(np.c_[xx.ravel(), yy.ravel()])
```

```
# Plotting decision boundary and margins
Z = Z.reshape(xx.shape)
plt.contour(xx, yy, Z, colors='k', levels=[-1, 0, 1], alpha=0.5, linestyles=
plt.show()
```



In [32]:
```
# Plotting decision boundary with support vectors
support_vectors = svm.support_vectors_
plt.scatter(X[:, 0], X[:, 1], c=y, cmap='viridis')
plt.scatter(support_vectors[:, 0], support_vectors[:, 1], color='red', marke
plt.title('SVM Decision Boundary with Support Vectors')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.legend()
plt.show()
```


SVM Decision Boundary with Support Vectors
Support Vectors
Feature 2
Feature 1
3
2
1
0
−1
−2
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0

```
In [ ]:
```