Introduction

The vertebral column dataset from UC Irvine's ML repository contains values for six biomechanical features used to classify orthopedic patients into 2 classes (normal or abnormal) or 3 classes (normal, disk hernia or spondylolisthesis)[1].

EDA and Preprocessing

Consisting of 310 entries (210 abnormal and 100 normal), the dataset had no missing values. Initial exploration identified several (mostly reasonable) outliers across variables, with one particular point showing unusually high values for 'pelvic_incidence', 'sacral_slope', and 'grade_of_spondylolisthesis'[2], suggesting an extreme anomaly or error, leading to its removal.

Further EDA revealed correlations and highlighted potential relationships. The distribution of 'grade_of_spondylolisthesis' was notably skewed right. I experimented with several transformation methods and settled on Yeo-Johnson transformation[3] as it made the distribution look the most normal.

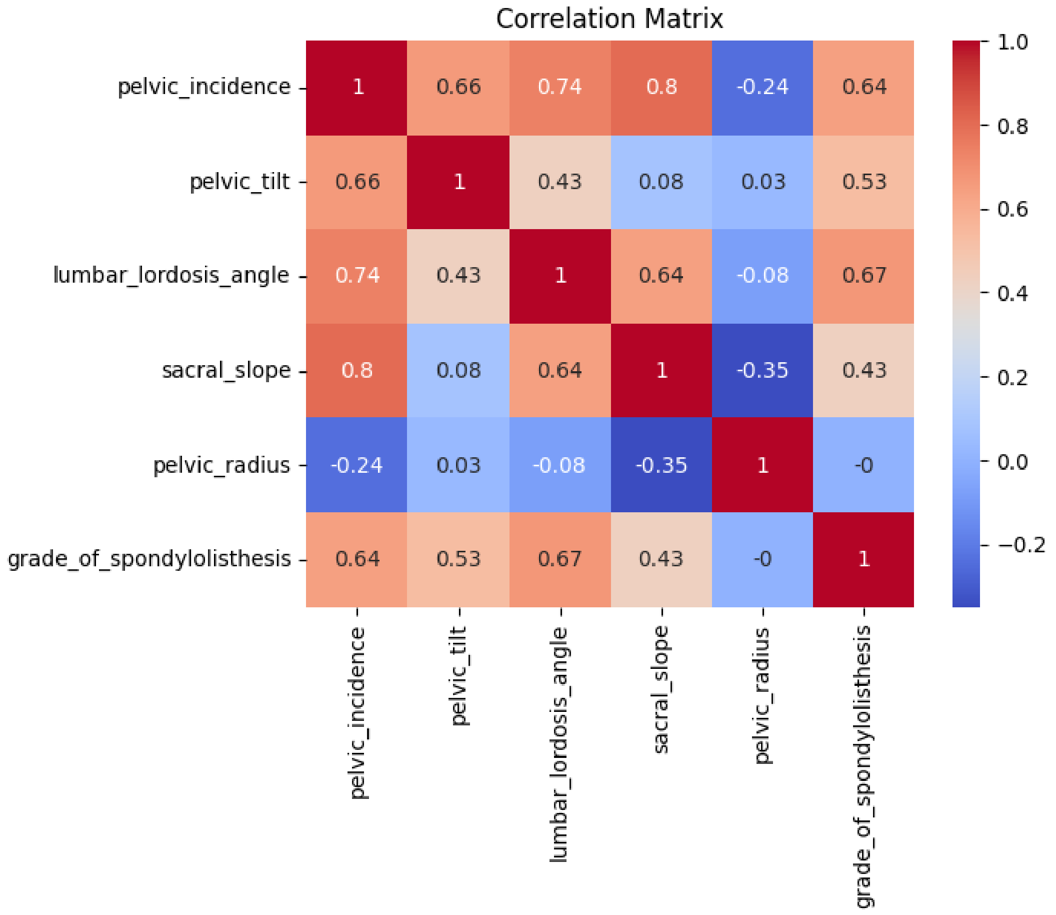


Figure 1- Correlation Matrix

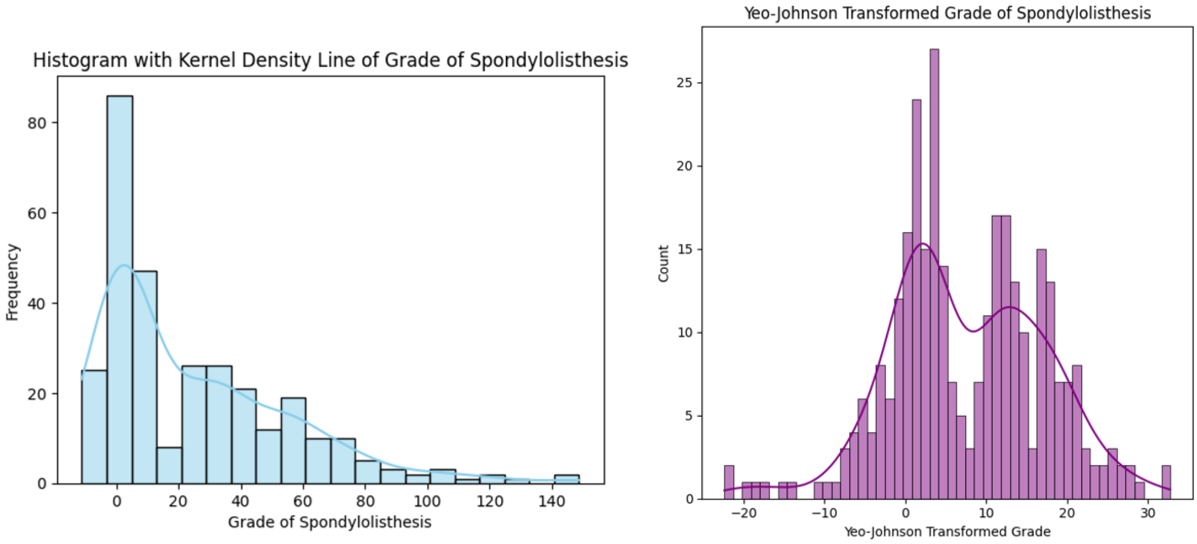


Figure 2- Grade of Spondylolisthesis before (left) and after Yeo-Johnson Transformation

After standardising the data, I applied Principal Component Analysis (PCA) to reduce dimensionality. I opted for two principal components (PCs) for visualisation of model results and decision boundaries. PC1 and PC2 explained 75.76% of the data variance.

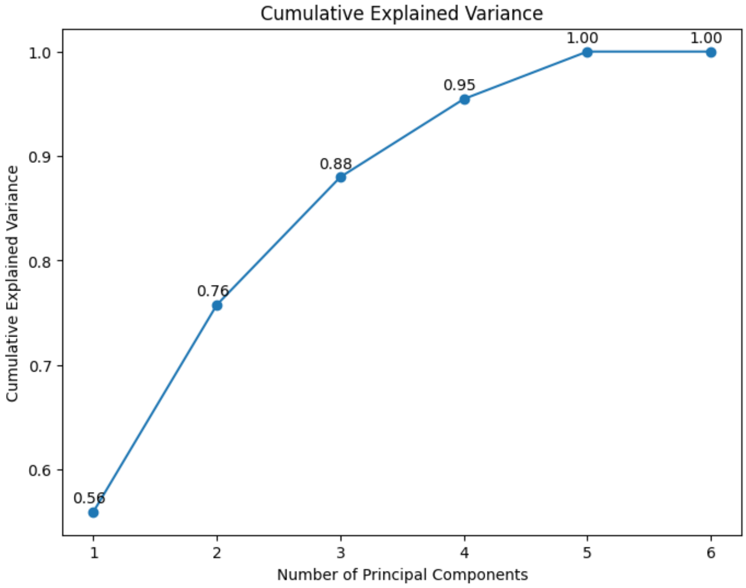


Figure 3- Number of PCs vs Cumulative Explained Variance

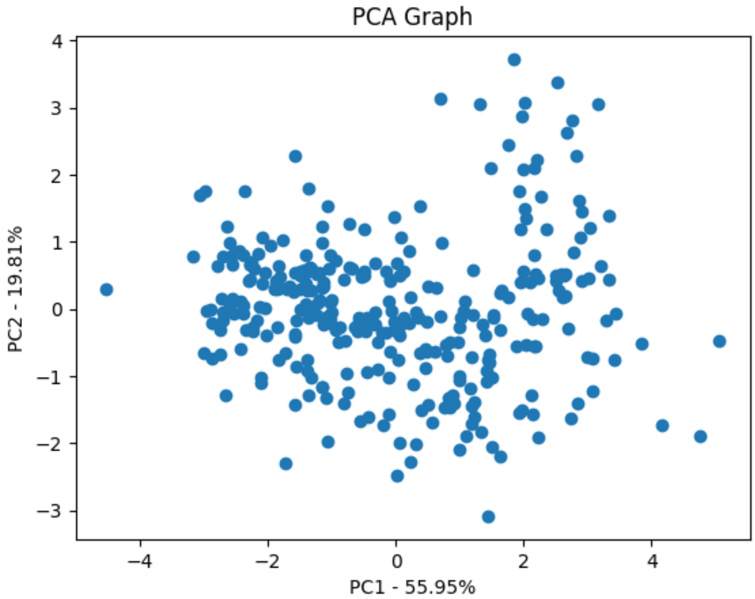


Figure 4- PCA Plot

Unsupervised Learning

My choice, k-means clustering, required me to specify the number of clusters. An elbow plot confirmed my understanding of having 2 or 3 clusters based on the dataset's background. The plot indicated a substantial reduction in the within-cluster sum of squares when increasing the number of clusters from 1 to 2 and 2 to 3, suggesting the presence of distinct groups. This report specifically focuses on distinguishing between normal and abnormal classes.

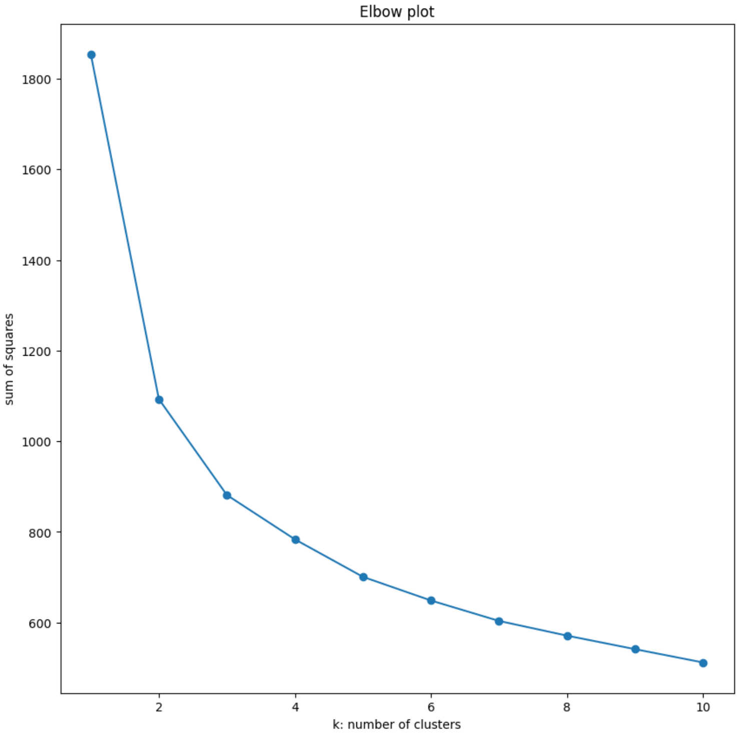


Figure 5- Elbow Plot

K-means clustering aims to minimise the sum of squares within each cluster. The sum of squares is defined as the square of the distance between each data point and the centre of the cluster. Mathematically, the goal is:

$$min_C \sum_{j=1}^{k} \sum_{x \in C_j} ||x - \mu_j||_2^2$$

where j is the number of clusters and x is the datapoint in each cluster C, and μ_j is the centre of cluster j, defined as:

$$\mu_j = \frac{1}{|C_j|} \sum_{x \in C_j} x$$

To determine the distance, Euclidian distance (shown below) is used:

d(x, μ) = | | $x - \mu$ | |$_2$

where | |u| |$_2$ = $\sqrt{\sum_{j=1}^{d} u_j^2}$ = $\sqrt{u_1^2 + u_2^2 + \ldots + u_d^2}$

Informally, the algorithm works like this:

1. Choose initial positions for the cluster centres, μ_j (randomly or using an initialisation method).
2. Each data point in the dataset is assigned to the nearest centroid (Euclidian distance).
3. The centroids, μ'_j, are recalculated after assignment of all data points.
4. Repeat 2 and 3 until the centres stop moving for every cluster (when $\mu_j = \mu'_j$).

Supervised Learning

Since the two classes were imbalanced, I under-sampled the larger class (AB), resulting in a balanced dataset with 100 'AB' and 100 'NO' observations. This would ensure the algorithm predicts both classes equally well.

My chosen algorithm, RandomForest classifier, uses multiple decision trees to predict a class and selects the mode of the individual trees' predictions as output. Here is how it works:

1. Bootstrap Sampling- For each tree $T_{i,}$ a random subset of training data D_i is sampled **with replacement** from the original dataset D.
2. Feature Randomisation- At each split of the decision tree, a random subset of features F_i is considered.
3. Classifying- Each tree predicts a class for each new data point x, and the final output is determined by a majority vote (mode).

This method mitigates overfitting by combining multiple trees, trained on different subsets of data. That said, I still limited 'max_depth' to 2 to minimise overfitting. RandomForest is also adept at managing irrelevant features and multicollinearity, making it suitable for datasets with complex feature interactions.

My chosen library, scikit-learn, uses Gini index/impurity as a criterion for splitting nodes[4]. The Gini index is defined as:

Gini(D) = 1- $\sum_{i=1}^{k} p_i^2$

where D is the dataset, k is the number of classes, and p_i is the proportion of observations in D that belong to class i[5].

The goal at each split is to minimise this Gini index, with the ideal scenario being all observations at a node belonging to a single class (Gini index of 0). A completely balanced node, where classes are evenly split, would exhibit a Gini index of 0.

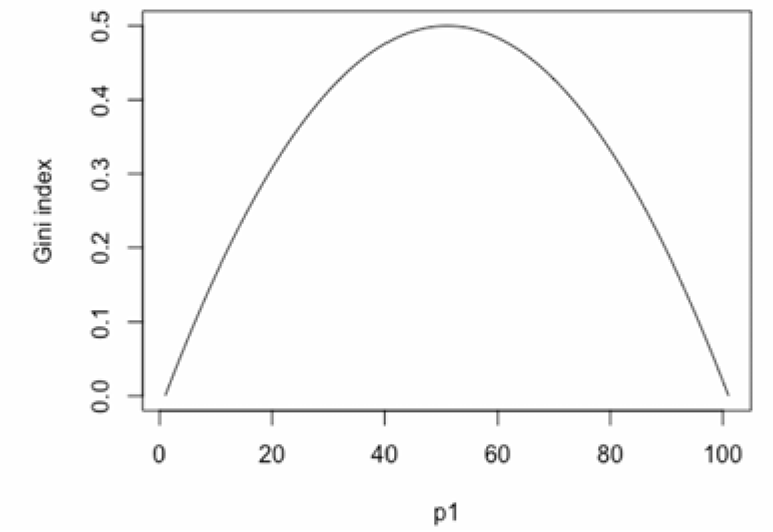


Figure 6- Proportion of p1 vs Gini Index

Results and Analysis

The actual classes/clusters show noticeable differences with some overlap (Figure 8). This highlights the limitations of k-means clustering (Figure 9), which assumes spherical clusters of similar size. The primary issue arises from its inability to accommodate clusters with varying shapes and densities, as observed in our scenario.

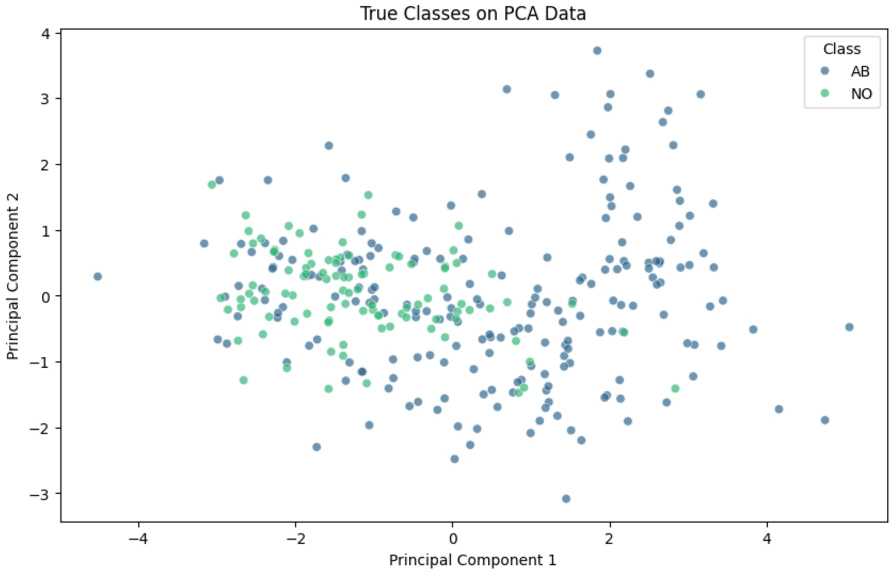


Figure 7- True Classes

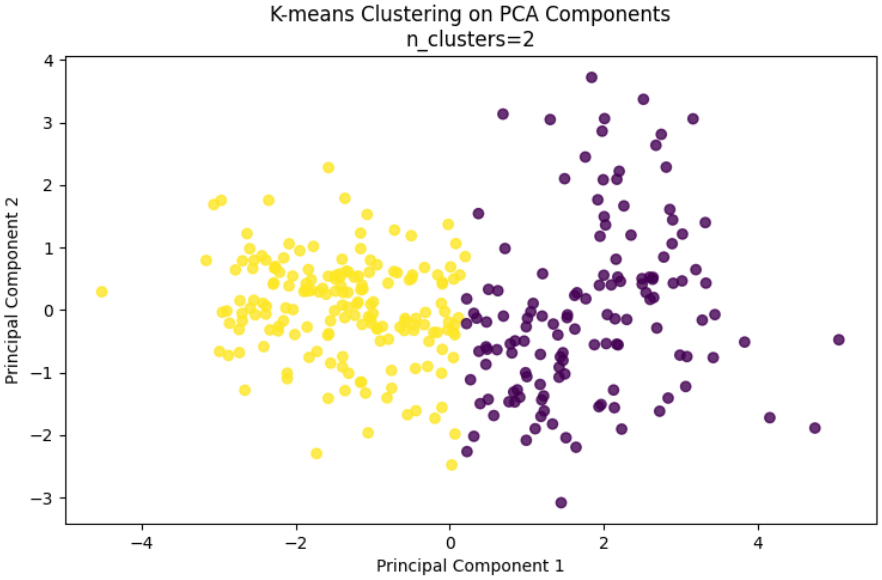


Figure 8- K-means Clustering, showing a somewhat linear division

For supervised learning, I also evaluated SVM (Radial Basis Function) and KNN in addition to RandomForest using an 80-20 train-test split with PCA and standardisation applied post-split to avoid data leakage. Models were assessed under three scenarios, using:

- Only PC1 and PC2,
- All 6 PCs,
- Scaled data without PCA.

I used F-1 score for evaluation as the validation dataset has imbalanced classes.

Table 1 and 2- Comparison of Different Models

Models’ F-1 Score for Each Class (Training Data)

	PC1 and PC2		6 PCs		Scaled Data	
	AB	NO	AB	NO	AB	NO
SVM	0.77	0.80	**0.87**	**0.88**	**0.87**	**0.88**
RandomForest	0.78	0.81	0.86	0.88	0.80	0.83
KNN	**0.79**	**0.81**	0.85	0.86	0.85	0.86

Models’ F-1 Score for Each Class (Validation Data)

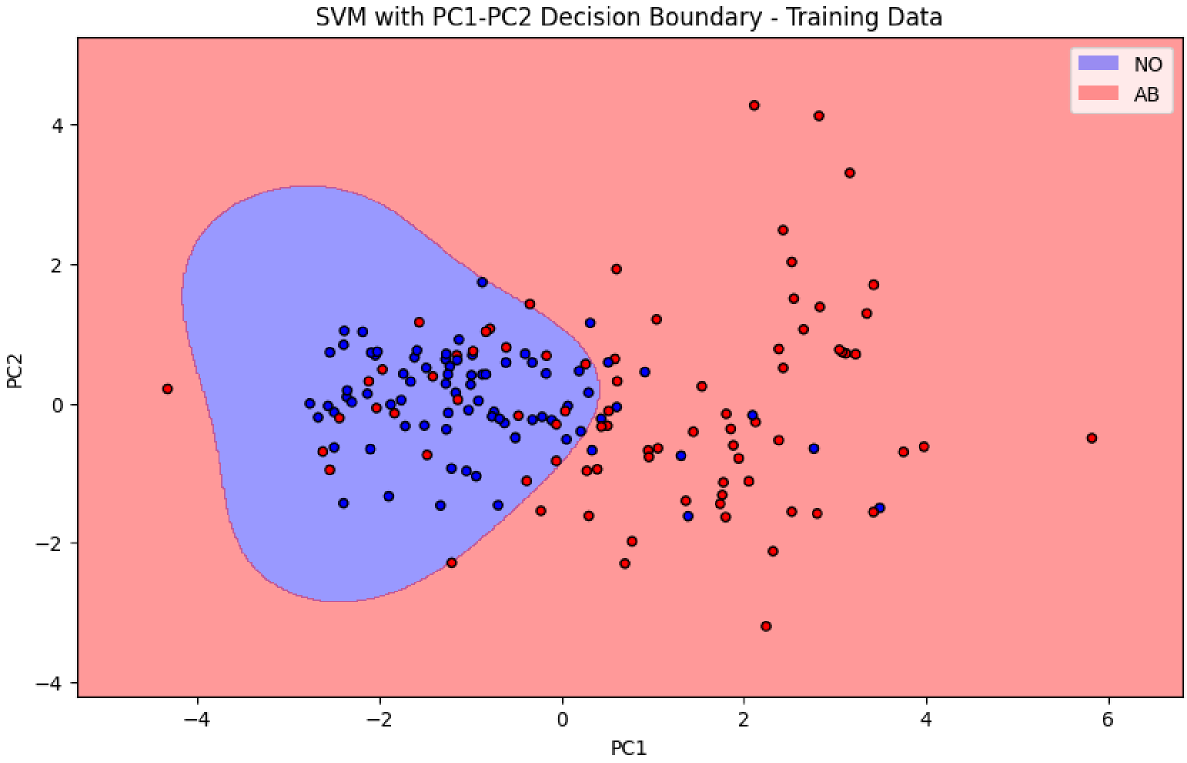
	PC1 and PC2		6 PCs		Scaled Data	
	AB	NO	AB	NO	AB	NO
SVM	0.80	0.62	**0.93**	**0.84**	**0.93**	**0.84**
RandomForest	**0.80**	**0.64**	0.84	0.68	0.87	0.73
KNN	0.79	0.60	0.88	0.75	0.88	0.75

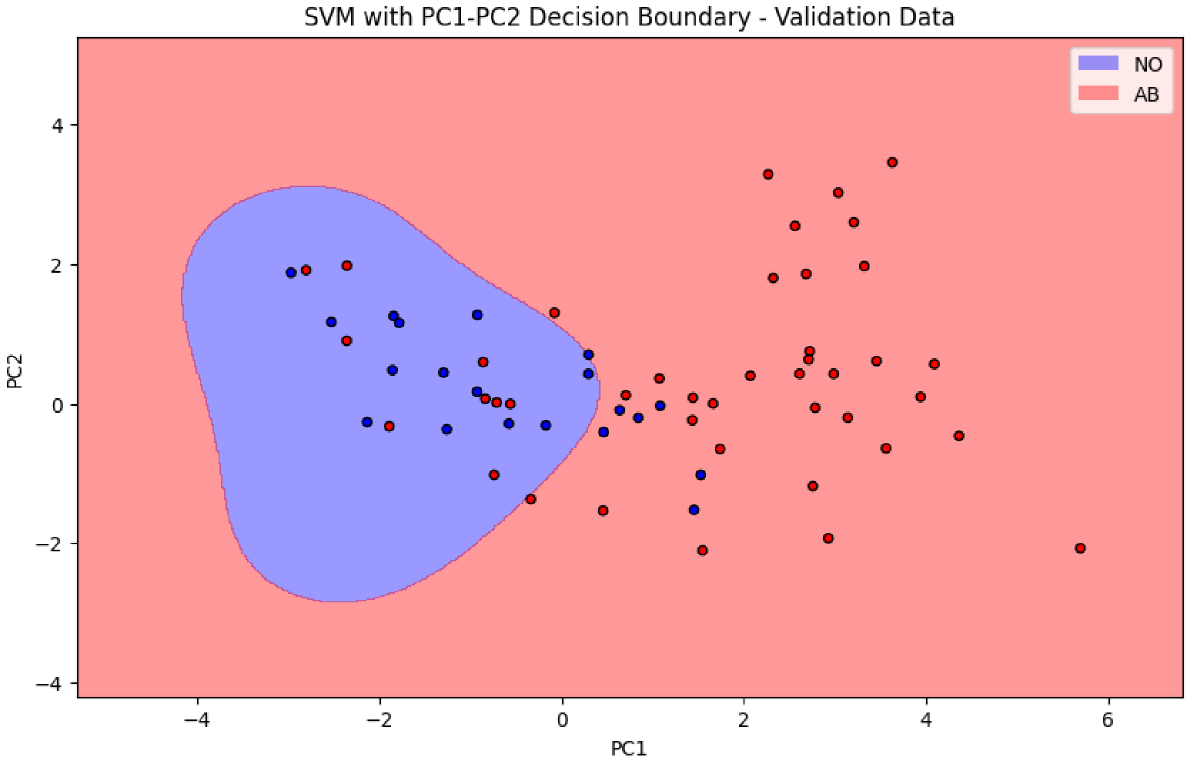
Using only PC1 and PC2 allowed me to visualise the decision boundaries and performance (Figure 9) on validation data. However, the tradeoff was reduced performance due to reduced total explained variance.

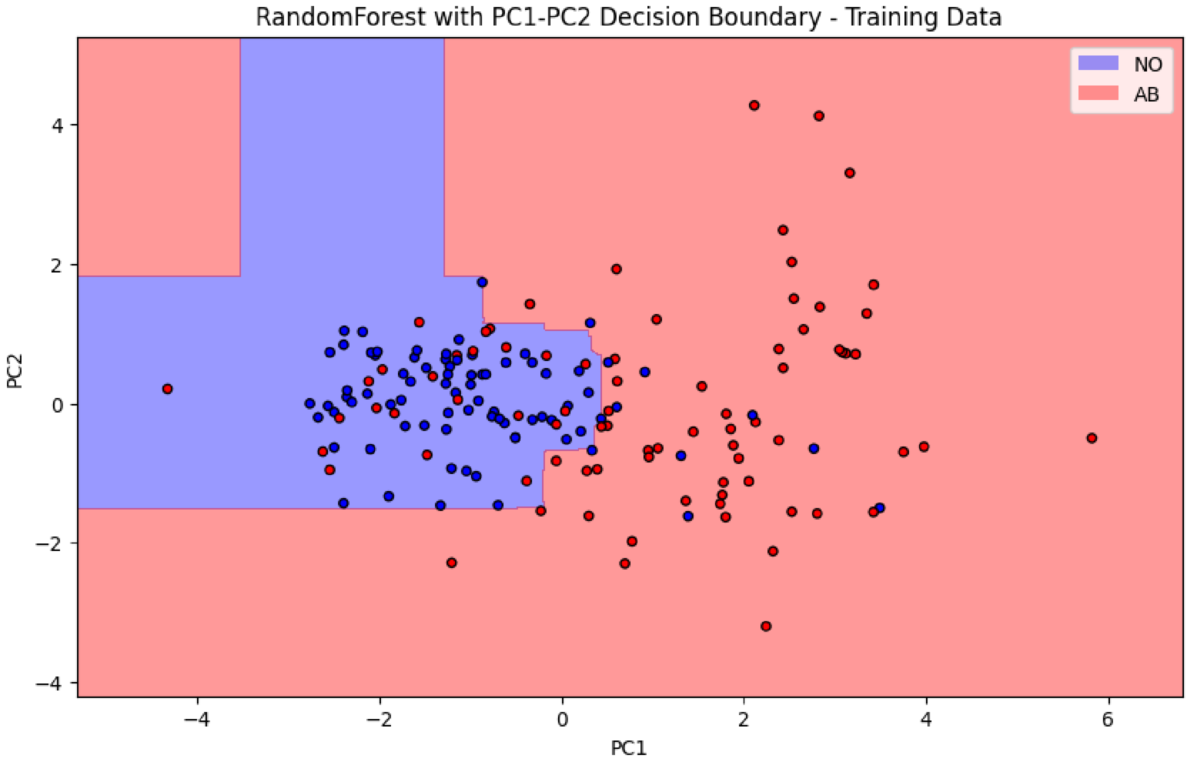
SVM outperformed others in validation, except when based solely on PC1 and PC2. Training data showed higher F-1 scores for "NO", but this trend reversed in validation data because while the training set was balanced, the validation set preserved the original imbalance, providing more opportunities to predict “AB” correctly. Moreover, the supervised methods’ boundaries favoured ‘AB’, giving it a much wider region.

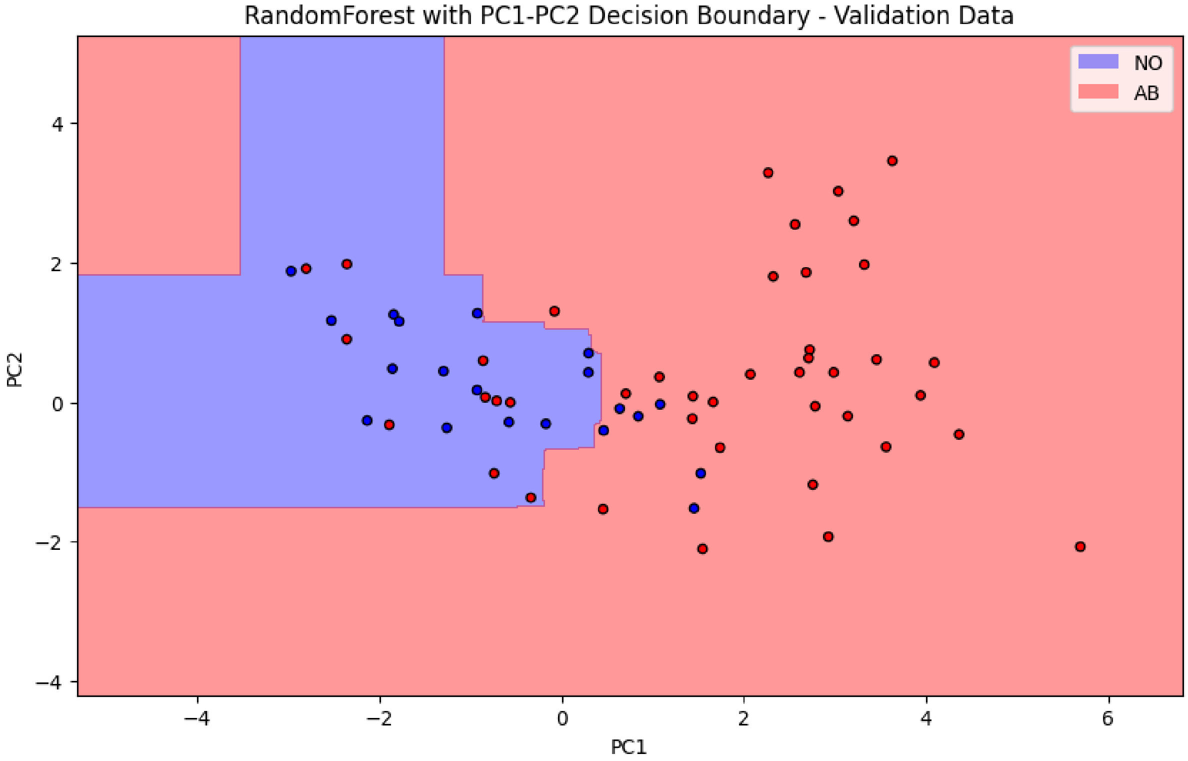
K-means clustering’s linear division suggests oversimplification, missing complex nonlinear relationships captured by supervised methods. SVM, in particular, excels using its kernel trick, finding hyperplanes in higher-dimensional spaces without explicitly computing data coordinates.

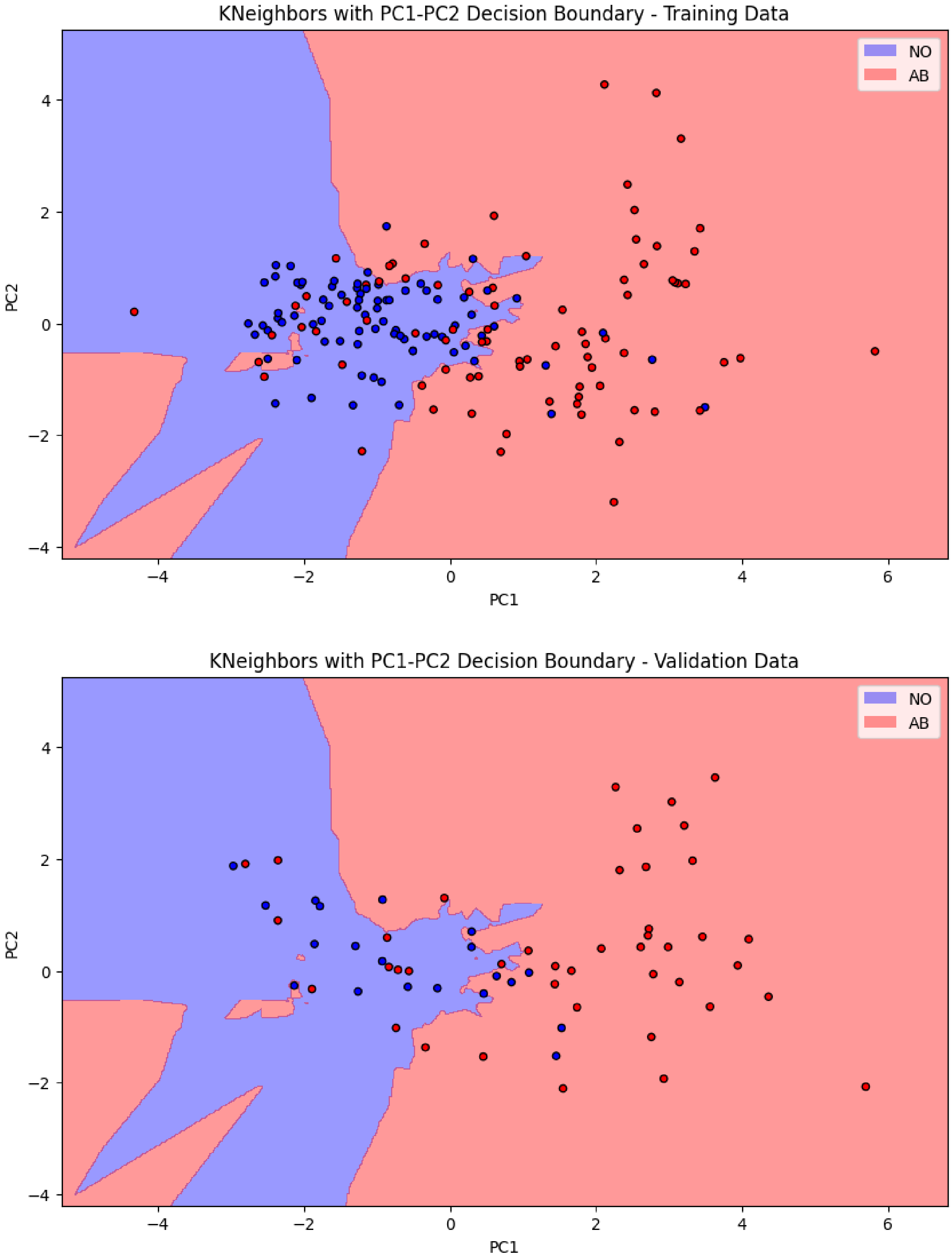
In conclusion, the supervised methods performed much better than the unsupervised counterpart. However, the imbalance in prediction power across the two classes persists across models. Therefore, if ‘NO’ class is important, alternative balancing techniques, employing other dimensionality-reduction methods, or testing additional models may be required to achieve a more balanced result.


SVM with PC1-PC2 Decision Boundary - Training Data
NO
AB
PC1
PC2
−4
−2
0
2
4
6
−4
−2
0
2
4


SVM with PC1-PC2 Decision Boundary - Validation Data
NO
AB
PC1
PC2
−4
−2
0
2
4
6
−4
−2
0
2
4


RandomForest with PC1-PC2 Decision Boundary - Training Data
NO
AB
PC1
PC2
−4
−2
0
2
4
6
−4
−2
0
2
4


RandomForest with PC1-PC2 Decision Boundary - Validation Data
NO
AB
PC1
PC2
−4
−2
0
2
4
6
−4
−2
0
2
4


KNeighbors with PC1-PC2 Decision Boundary - Training Data

KNeighbors with PC1-PC2 Decision Boundary - Validation Data

Figure 9- Performance and Decision Boundaries of Each Model

Bibliography

1. Barreto G, Neto A. Vertebral Column. archive.ics.uci.edu. Published August 8, 2011. Accessed March 6, 2024. https://archive.ics.uci.edu/dataset/212/vertebral+column

2. Le Huec JC, Aunoble S, Philippe L, Nicolas P. Pelvic parameters: origin and significance. *European spine journal : official publication of the European Spine Society, the European Spinal Deformity Society, and the European Section of the Cervical Spine Research Society*. 2011;20(S5):564-571. doi:https://doi.org/10.1007/s00586-011-1940-1

3. Weisberg S. *Yeo-Johnson Power Transformations*.; 2001. Accessed March 6, 2024. https://www.stat.umn.edu/arc/yjpower.pdf

4. Scikit-learn. sklearn.ensemble.RandomForestClassifier — scikit-learn 0.20.3 documentation. Scikit-learn.org. Published 2018. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

5. Dash S. Decision Trees Explained — Entropy, Information Gain, Gini Index, CCP Pruning.. Medium. Published November 2, 2022. Accessed March 6, 2024. https://towardsdatascience.com/decision-trees-explained-entropy-information-gain-gini-index-ccp-pruning-4d78070db36c