REPORT ON PARKINSON PREDICTION MODEL

Shreya Mishra

SHREYAMISHRA414@GMAIL.COM

Table of Contents

Report on Parkinson's Prediction Model

This project focuses on a classification model. We have applied different Machine Learning models to predict the presence of Parkinson's disease in a patient.

Parkinson's disease

Parkinson's disease is a progressive disorder that affects the nervous system and the parts of the body controlled by the nerves. Symptoms start slowly. The first symptom may be a barely noticeable tremor in just one hand. Tremors are common, but the disorder may also cause stiffness or slowing of movement. Although Parkinson's disease can't be cured, medications might significantly improve your symptoms. Occasionally, your health care provider may suggest surgery to regulate certain regions of your brain and improve your symptoms.

Citation for the dataset

'Exploiting Nonlinear Recurrence and Fractal Scaling Properties for Voice Disorder Detection',

Little MA, McSharry PE, Roberts SJ, Costello DAE, Moroz IM.

BioMedical Engineering OnLine 2007, 6:23 (26 June 2007)

License

Dataset

The dataset is available at Kaggle

https://www.kaggle.com/datasets/gargmanas/parkinsonsdataset

Dataset Description

The dataset consists of 24 columns and 195 records.

The dataset contains 23 attributes and 1 target variable.

Attribute Information

1. name - ASCII subject name and recording number

2. MDVP:Fo(Hz) - Average vocal fundamental frequency

3. MDVP:Fhi(Hz) - Maximum vocal fundamental frequency

4. MDVP:Flo(Hz) - Minimum vocal fundamental frequency

5. MDVP:Jitter(%), MDVP:Jitter(Abs), MDVP:RAP, MDVP:PPQ, Jitter:DDP - Several measures of variation in fundamental frequency

6. MDVP:Shimmer, MDVP:Shimmer(dB), Shimmer:APQ3, Shimmer:APQ5, MDVP:APQ, Shimmer:DDA - Several measures of variation in amplitude

7. NHR, HNR - Two measures of ratio of noise to tonal components in the voice

8. status - Health status of the subject (one) - Parkinson's, (zero) - healthy

9. RPDE, D2 - Two nonlinear dynamical complexity measures

10. DFA - Signal fractal scaling exponent

11. spread1, spread2, PPE - Three nonlinear measures of fundamental frequency variation

12. Tonnetz - The set of pitch classes used to characterize each note

Target Variable

status - Health status of the subject (one) - Parkinson's, (zero) - healthy

Models Used

1. Logistic Regression

2. Decision Tree

3. Pruned Decision Tree

4. Random Forest

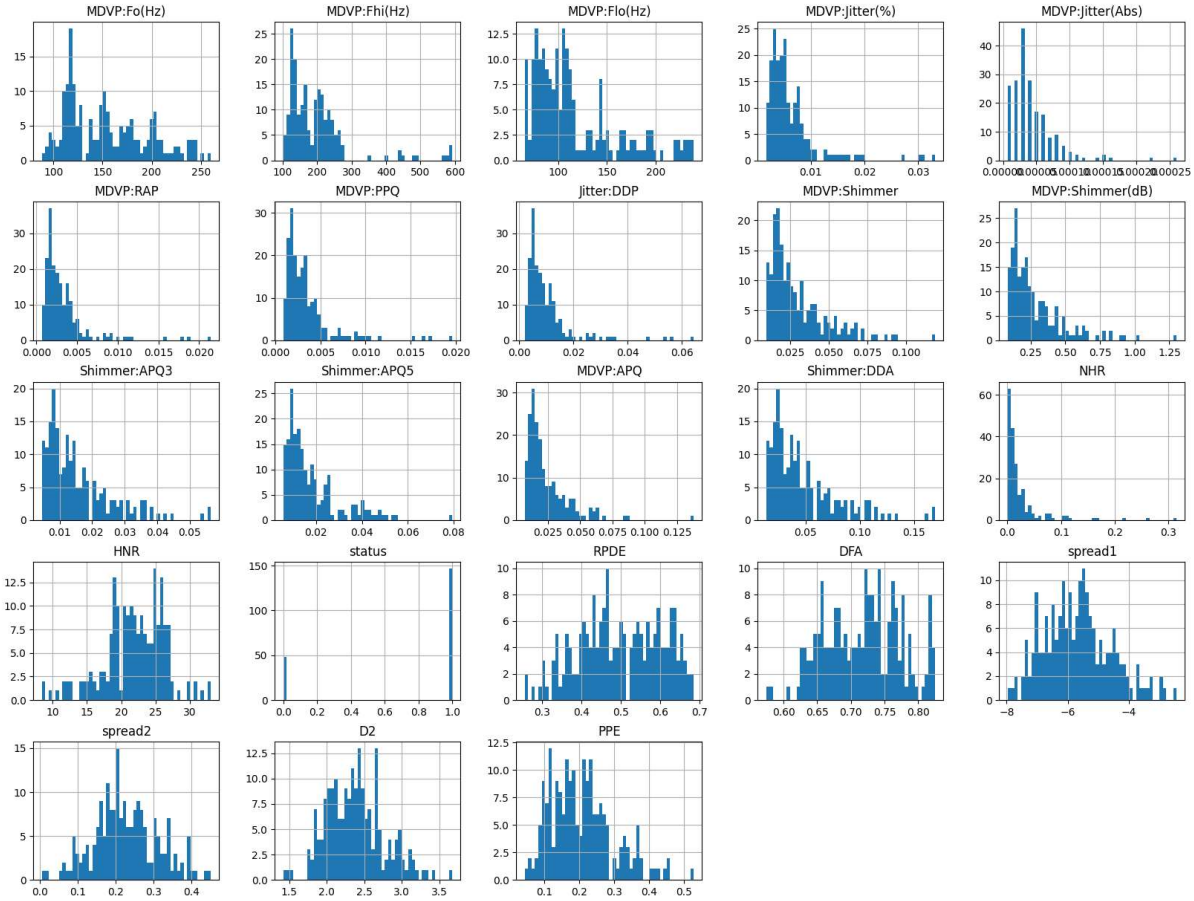
5. XGBClassifier

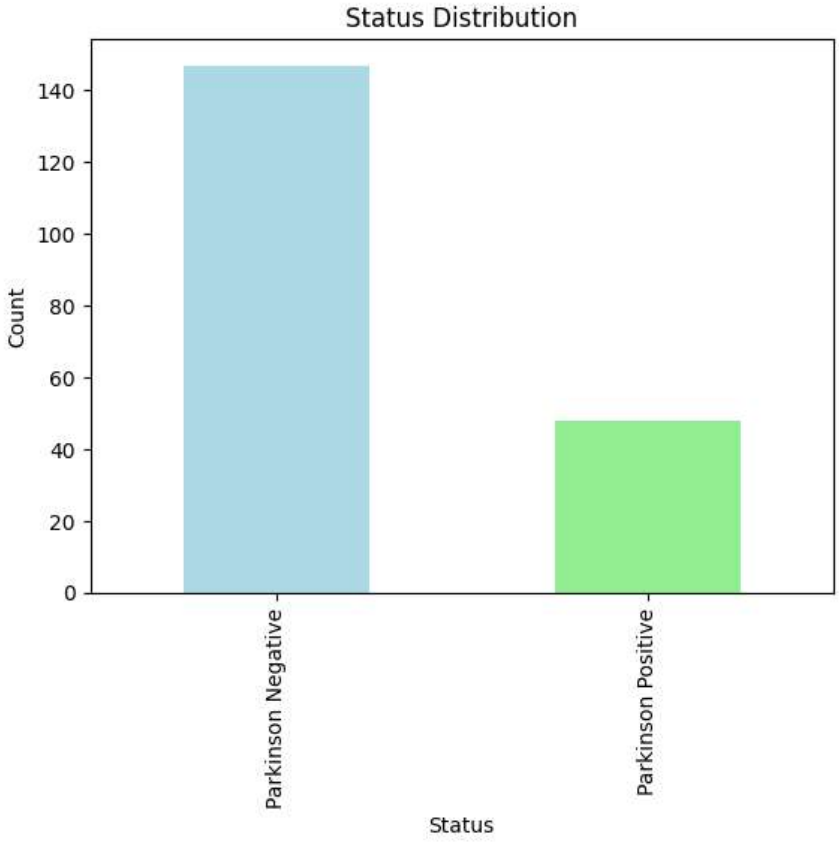
6. Support Vector Machine

Techniques used for Data Preprocessing

We have 'name', 'MDVP:Fo(Hz)', 'MDVP:Fhi(Hz)', 'MDVP:Flo(Hz)', 'MDVP:Jitter(%)', 'MDVP:Jitter(Abs)', 'MDVP:RAP', 'MDVP:PPQ', 'Jitter:DDP', 'MDVP:Shimmer', 'MDVP:Shimmer(dB)', 'Shimmer:APQ3', 'Shimmer:APQ5', 'MDVP:APQ', 'Shimmer:DDA', 'NHR', 'HNR', 'status', 'RPDE', 'DFA', 'spread1', 'spread2', 'D2', 'PPE' as our columns in parkinsons.csv file. We have dropped the ‘name’ column from the dataframe. We see that all the variables except ‘status’ are continuous numerical variables. The ‘status’ is a categorical variable with values 1 and 0.

1- Parkinson Positive
0- Parkinson Negative





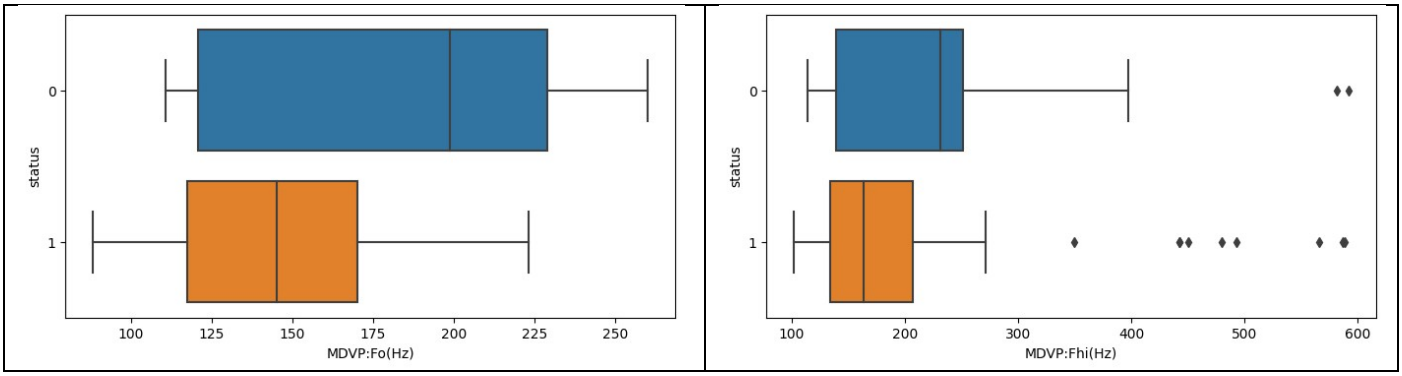
From the above diagram, we can conclude that our dataset is imbalanced. The people with Parkinson positives are only 48 in number out of 195. Thus, when evaluating the model's performance, accuracy alone may not be an appropriate metric.

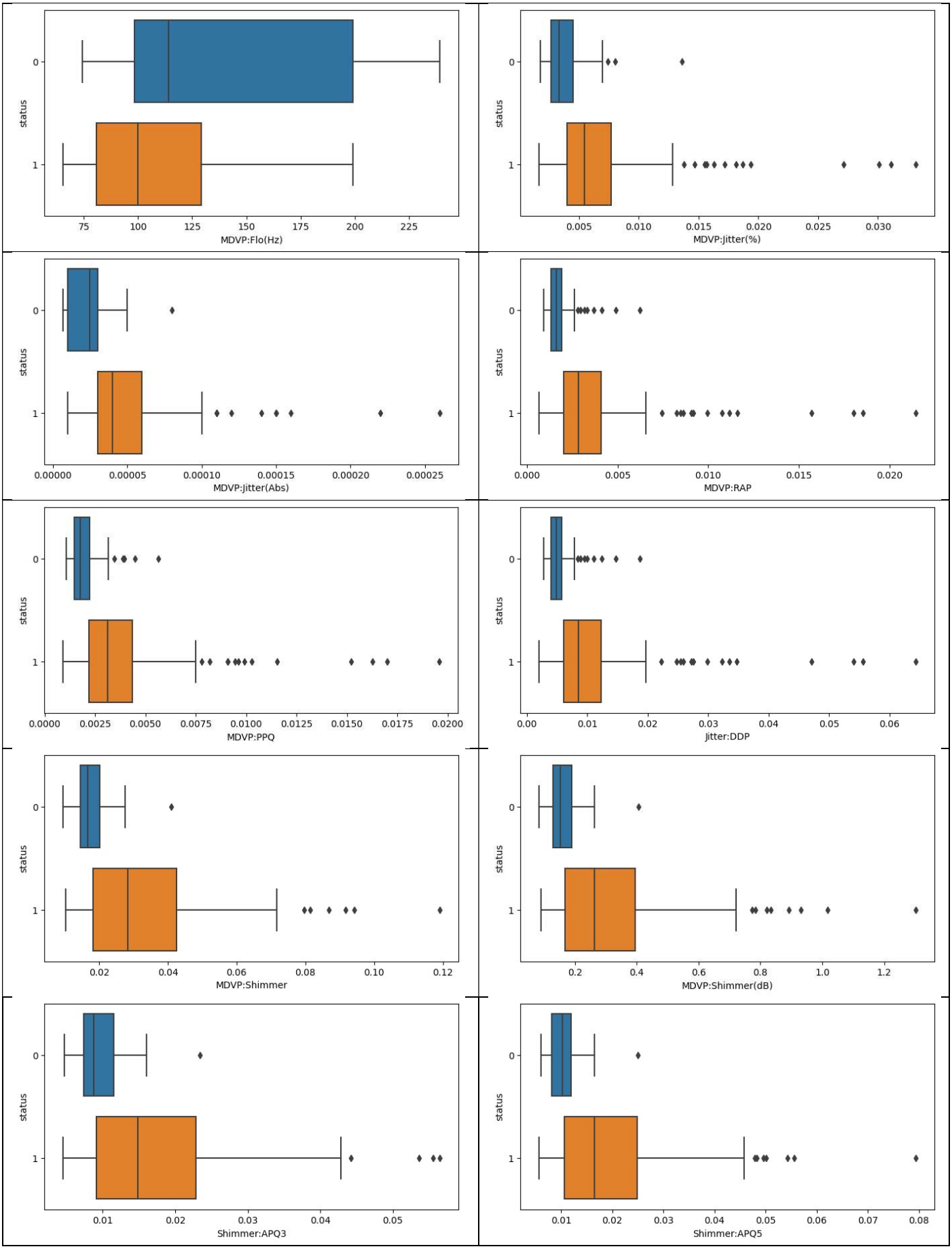
Imbalanced datasets can lead to a bias in the trained model towards the majority class. In this case, the model may perform well at identifying individuals without Parkinson's disease but poorly at identifying those with the disease, which could be bad.

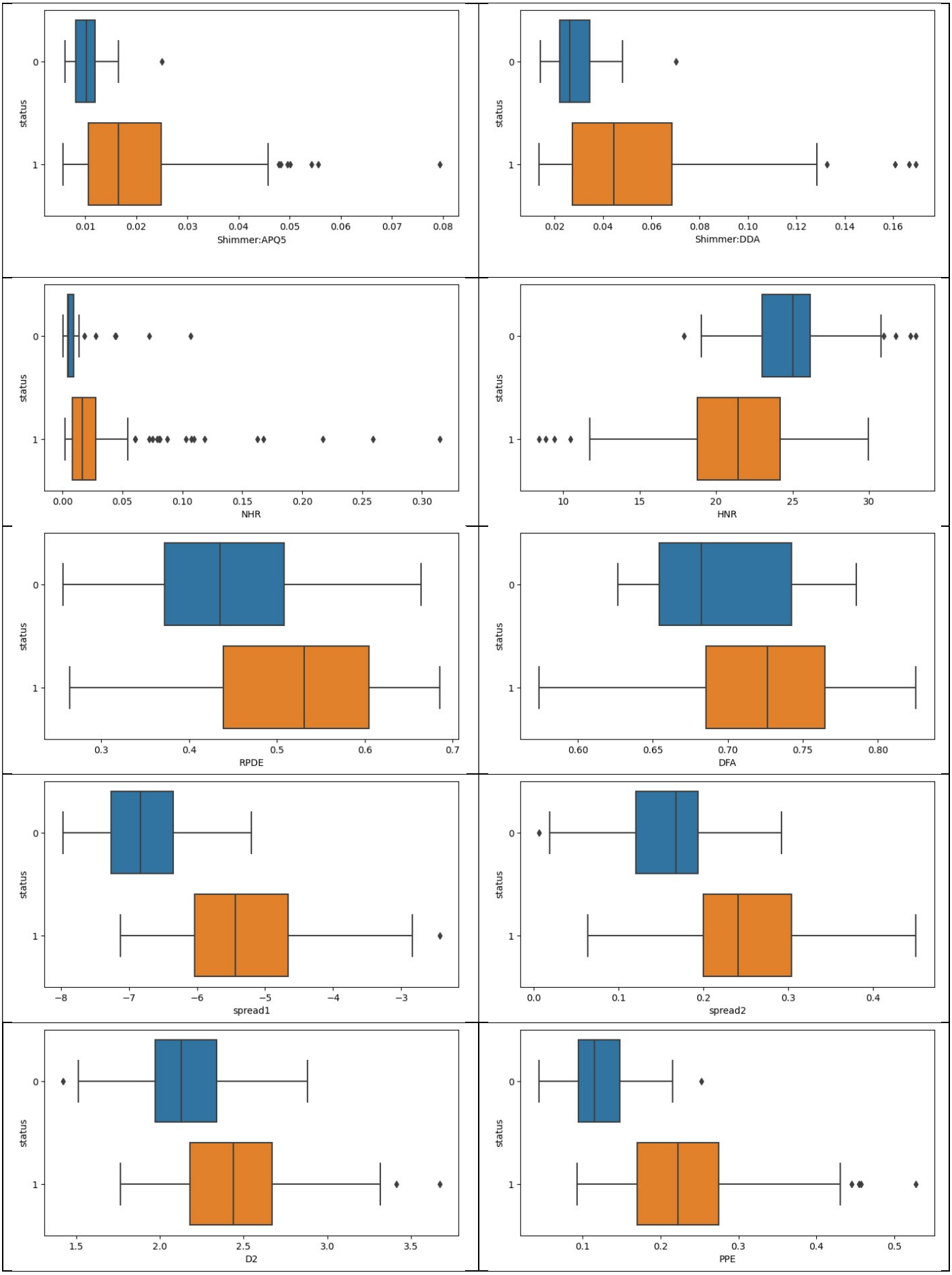
We might need to employ resampling techniques to address the class imbalance. These techniques include oversampling the minority class, undersampling the majority class, or using more advanced methods like Synthetic Minority Over-sampling Technique (SMOTE) to generate synthetic samples. We have used SMOTE technique later in the model training.

Machine learning algorithms may be more sensitive to class imbalance than others. For instance, decision trees can struggle with imbalanced data, while ensemble methods like Random Forests or Gradient Boosting can handle it better.

Distribution of Variables among people who have Parkinson vs those who do not

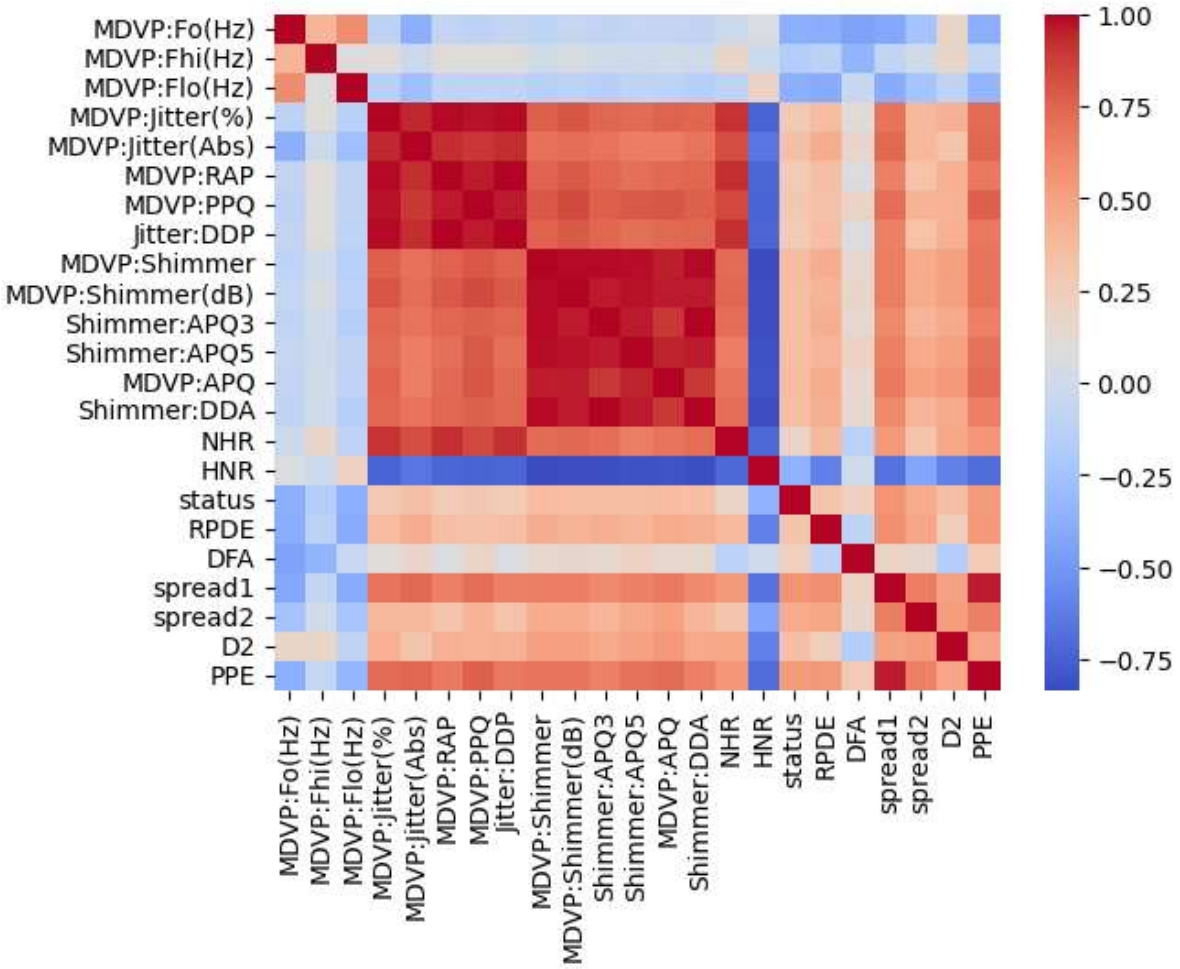



status
0
1
MDVP:Flo(Hz)
75 100 125 150 175 200 225
MDVP:Jitter(%)
0.005 0.010 0.015 0.020 0.025 0.030
MDVP:Jitter(Abs)
0.00000 0.00005 0.00010 0.00015 0.00020 0.00025
MDVP:RAP
0.000 0.005 0.010 0.015 0.020
MDVP:PPQ
0.0000 0.0025 0.0050 0.0075 0.0100 0.0125 0.0150 0.0175 0.0200
Jitter:DDP
0.00 0.01 0.02 0.03 0.04 0.05 0.06
MDVP:Shimmer
0.02 0.04 0.06 0.08 0.10 0.12
MDVP:Shimmer(dB)
0.2 0.4 0.6 0.8 1.0 1.2
Shimmer:APQ3
0.01 0.02 0.03 0.04 0.05
Shimmer:APQ5
0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08



From the above visualizations, we can see that the variables are skewed and of different summary statistics when one has Parkinson vs when one does not. The difference in ranges of values such as mean, median, min value, max value, etc can help us determine whether a person has Parkinson or not.

Feature Selection



The correlation matrix above shows that we have a few variables which are strongly correlated among each other.

Removing highly correlated variables, also known as feature selection or dimensionality reduction, can be beneficial for several reasons in data analysis and modeling:

1. Avoid Multicollinearity
2. Reduced Overfitting
3. Makes data difficult to interpret the impact of individual features on the target variable
4. Reduces Noise
5. Avoids Curse of Dimentionality

We define a threshold for correlation as 0.80. Any variable having correlation greater than 0.80 and less than -0.80 are removed

The columns which are dropped are(Feature Selection): -

HNR
Jitter:DDP
MDVP:APQ
MDVP:Jitter(Abs)
MDVP:PPQ
MDVP:RAP
MDVP:Shimmer(dB)
NHR
PPE
Shimmer:APQ3

Shimmer:APQ5
Shimmer:DDA

Oversampling using SMOTE

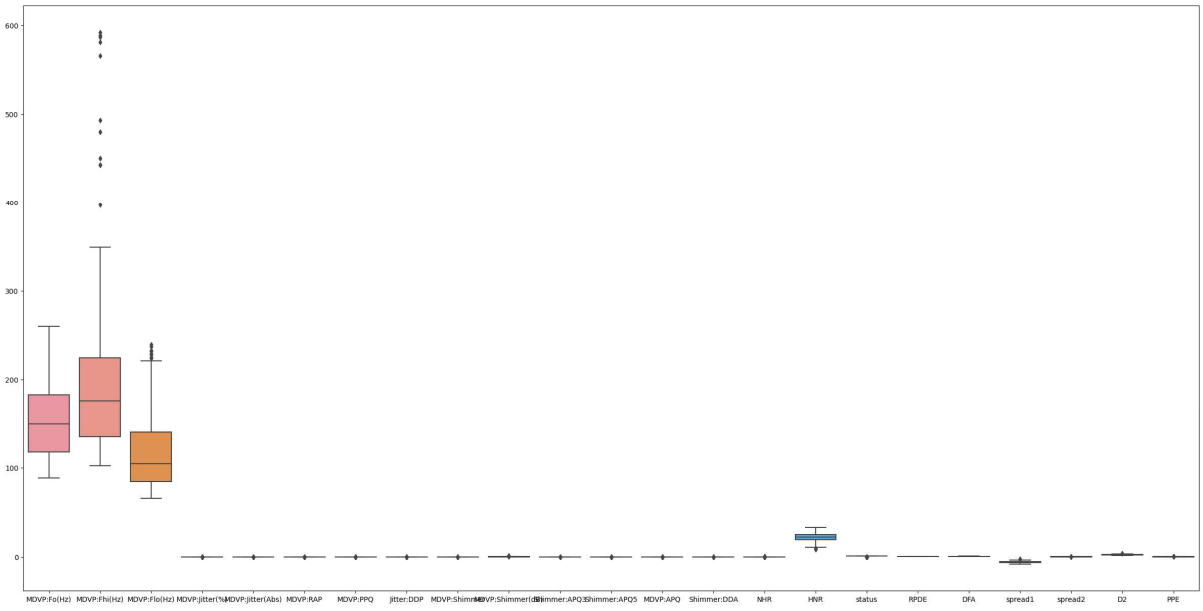
We can see that we have a class imbalance in our dataset. So, the models trained without Oversampling can lead to overfitting of the model. To avoid that, we use SMOTE (Synthetic Minority Over-sampling Technique) technique to generate synthetic samples for the minority class (those who have Parkinson's) by interpolating between existing samples. This helps balance the class distribution and prevents the model from being biased towards the majority class.

Class distribution after SMOTE:

1	147
0	147

In this way, we can balance our class. The other methods to achieve the same are ADASYN (Adaptive Synthetic Sampling), Borderline-SMOTE, Cluster-based Oversampling, etc.

Standardization of dataset



The boxplot displayed above demonstrates the requirement for standardisation of our data. We used the Standard Scalar Library for our standardisation technique, standardising the independent variables using the Z score.

Machine learning algorithms often make assumptions about the distribution of the data, assuming that features are centred around zero and have a similar scale. If our independent variables have different scales, some features may dominate the learning process, while others may have little influence.

Standardization transforms our data into a standard normal distribution with a mean of 0 and a standard deviation of 1. This makes it easier for machine learning algorithms to work effectively and can lead to better model performance.

The Z score is defined as

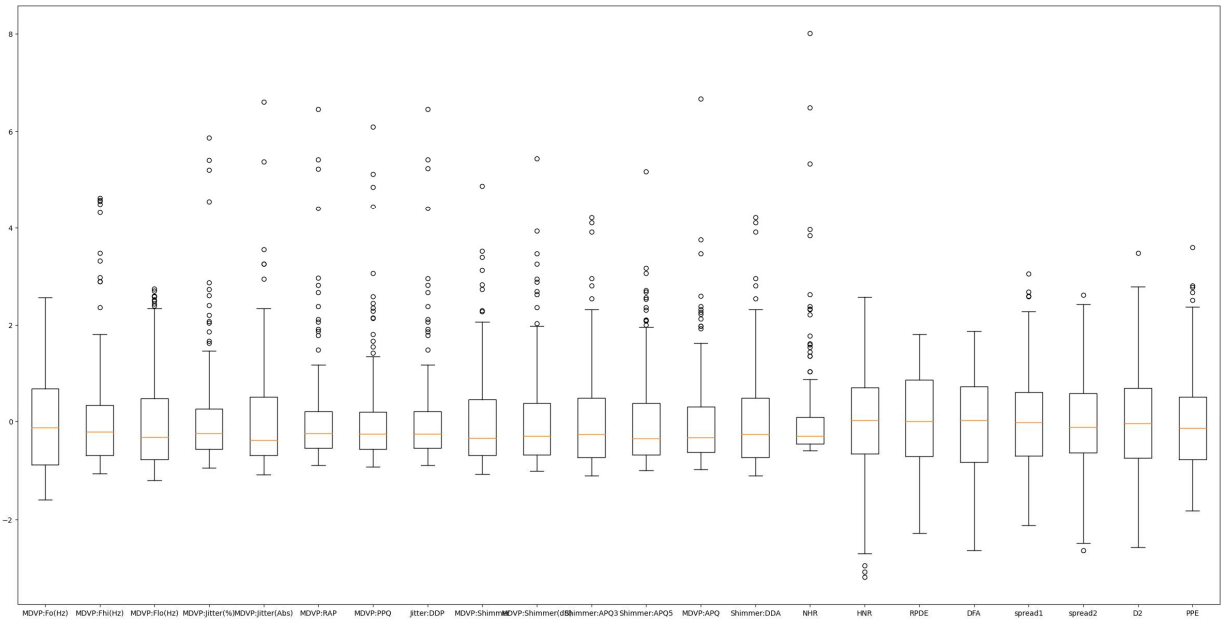
$z = (x - \mu) / \sigma$

1.

where:

z is the standardized value.
x is the original value.

μ is the mean of the variable.
σ is the standard deviation of the variable.



The data now looks standardized.

By applying standardization, you ensure that all your independent variables have similar ranges and distributions, which can help your machine learning models converge faster and produce more accurate results.

PCA

We have 22 independent variables in our dataset, but not all of them have been important and our model efficiency decreases due to curse of dimensionality.

Principal Component Analysis (PCA) is a dimensionality reduction technique that can help address the issue of having too many independent variables (features) in a dataset. It works by transforming the original features into a new set of linearly uncorrelated features called principal components, where the first principal component explains the most variance in the data, the second explains the second most variance, and so on. PCA can be used to reduce the dimensionality of your dataset while preserving as much information as possible.

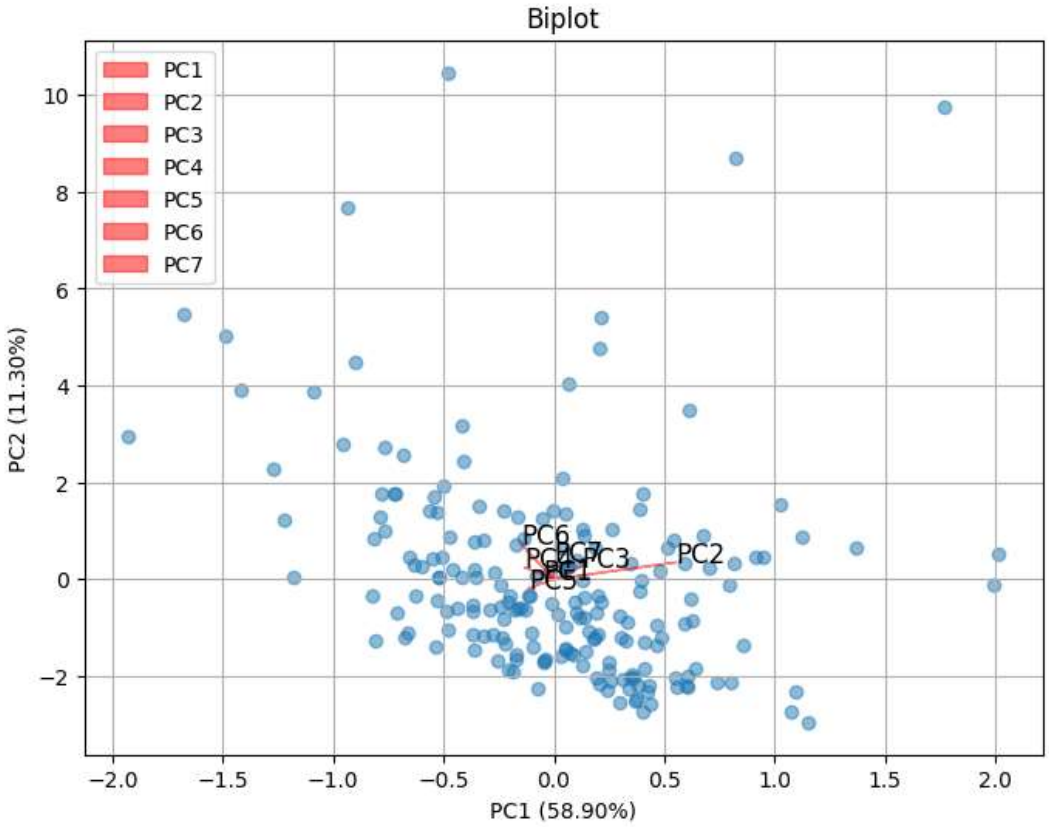
We initially take all 22 variables and make them undergo dimensionality reduction and transform the data into 22 Principle Components.

The eigen value for each component is given as: -

Singular Values (Eigenvalues): [5.02676002e+01 2.20169378e+01 1.73405825e+01 1.69018434e+01 1.37809157e+01 1.19237637e+01 1.03772710e+01 8.40646479e+00 7.51787458e+00 6.61094733e+00 5.23547546e+00 4.52151078e+00 3.68764168e+00 2.72808078e+00 2.07178187e+00 1.86240973e+00 1.55852417e+00 1.18606766e+00 8.25730369e-01 4.59963384e-01 8.39986608e-03 2.54148559e-03]

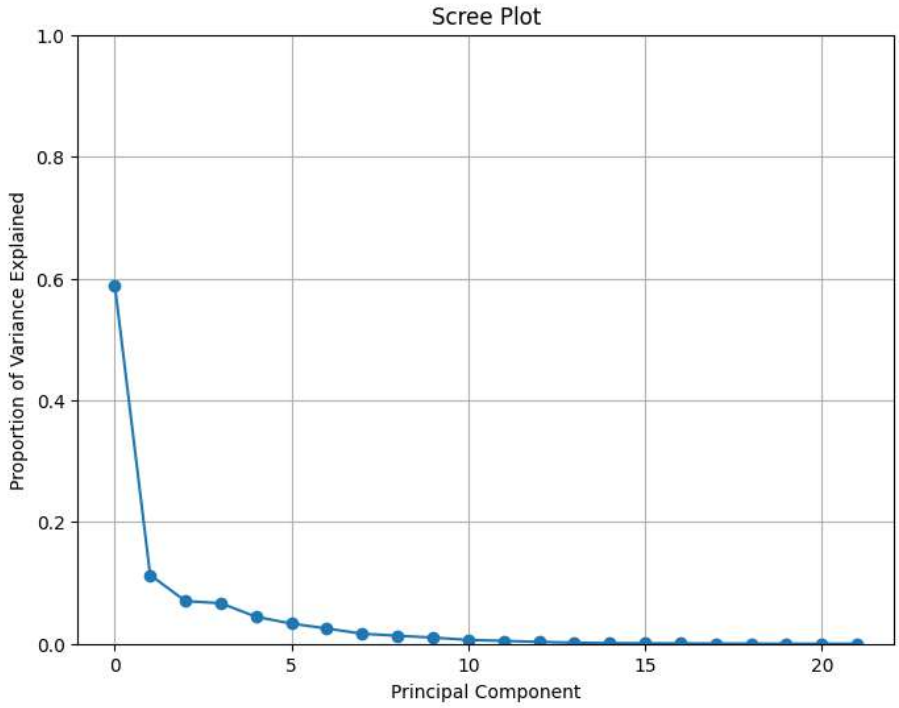
We also visualise a biplot. A biplot is a graphical representation of the results of a Principal Component Analysis (PCA) applied to a dataset. It displays both the principal components (often denoted as PC1 and PC2) and the original

variables in a single plot.

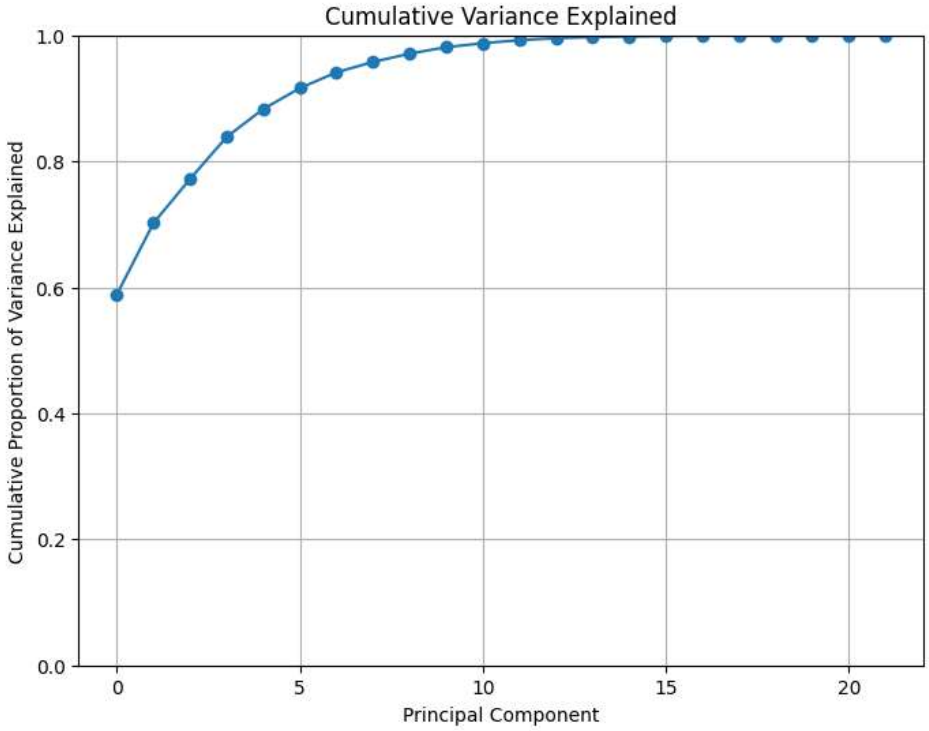


A scree plot is a graphical tool used in Principal Component Analysis (PCA) and factor analysis to help determine the number of principal components or factors to retain. It is a valuable visualization for understanding the variance explained by each component or factor and can assist in making informed decisions about dimensionality reduction.

- In PCA, each principal component represents a linear combination of the original variables and accounts for a certain amount of the total variance in the data.
- The scree plot displays the explained variance by each principal component in descending order.
- The "elbow" point in the scree plot is where the explained variance begins to level off. The number of components before this point is often chosen as the number of components to retain.



Cumulative explained variance represents the cumulative amount of variance explained by a subset of the principal components or factors. Calculating cumulative variance is useful for determining how much of the total variance in the dataset is captured as you retain an increasing number of components.



After all the above results, we go with 7 Principal components for our dataset as after 7 components, the variance explained is near to zero.

Analysis of Different Classification Models

Logistic Regression

Logistic regression is a statistical model used for binary classification and sometimes for multiclass classification problem. It uses the logistic function (also known as the sigmoid function) to map the linear combination of input features to a probability between 0 and 1.

The logistic (sigmoid) function is an S-shaped curve that smoothly transitions between 0 and 1. It is used to model the probability that a given input belongs to the positive class. The formula for the logistic function is:

$P(Y=1|X) = 1 / (1 + e^{(-z)})$

Where,
P(Y=1|X) is the probability of the positive class.
X represents the input features.
z is the linear combination of the input features and model parameters.

Technique 1: Feature Selection, Data Standardization

Train Data	**Test Data**
True Positive 113	True Positive 27
False Positive 13	False Positive 6
True Negative 23	True Negative 6
False Negative 7	False Negative 0
Accuracy: 0.8717948717948718	Accuracy: 0.8461538461538461
Precision: 0.8968253968253969	Precision: 0.8181818181818182
Recall: 0.9416666666666667	Recall: 1.0
F1_score: 0.9186991869918699	F1_score: 0.9
confusion matrix [[23 13] [7 113]]	confusion matrix [[6 6] [0 27]]

Technique 2: Feature Selection, Data Standardization, Oversampling

Train Data	**Test Data**
True Positive 100	True Positive 24
False Positive 24	False Positive 5
True Negative 92	True Negative 26
False Negative 19	False Negative 4
Accuracy: 0.8170212765957446	Accuracy: 0.847457627118644
Precision: 0.8064516129032258	Precision: 0.8275862068965517
Recall: 0.8403361344537815	Recall: 0.8571428571428571
F1_score: 0.8230452674897119	F1_score: 0.8421052631578947
confusion matrix [[92 24] [19 100]]	confusion matrix [[26 5] [4 24]]

Technique 3: Data Standardization, PCA

Train Data	**Test Data**
True Positive 108	True Positive 32
False Positive 17	False Positive 4
True Negative 24	True Negative 3
False Negative 7	False Negative 0
Accuracy: 0.8461538461538461	Accuracy: 0.8974358974358975
Precision: 0.864	Precision: 0.8888888888888888
Recall: 0.9391304347826087	Recall: 1.0
F1_score: 0.8999999999999999	F1_score: 0.9411764705882353
confusion matrix [[24 17] [7 108]]	confusion matrix [[3 4] [0 32]]

Technique 4: Oversampling, Data Standardization, PCA

Train Data	**Test Data**
True Positive 96	True Positive 25
False Positive 19	False Positive 5
True Negative 98	True Negative 25
False Negative 22	False Negative 4
Accuracy: 0.825531914893617	Accuracy: 0.847457627118644
Precision: 0.8347826086956521	Precision: 0.8333333333333334
Recall: 0.8135593220338984	Recall: 0.8620689655172413
F1_score: 0.8240343347639484	F1_score: 0.847457627118644
confusion matrix [[98 19] [22 96]]	confusion matrix [[25 5] [4 25]]

Model Evaluation

Based on both accuracy and recall, it appears that **Technique 3 (Data Standardization, PCA)** performs the best on the test data. It has the highest accuracy (0.8974) and recall (1.0), indicating that it correctly identifies all positive cases without making too many false positives.

Overfitting or Underfitting:

- Technique 1 shows good results but might be slightly overfitting on the training data as it has a higher recall on the training data compared to the test data.
- Technique 2 also shows signs of overfitting, as it has a lower recall on the test data compared to the training data.
- Technique 4 seems to be well-balanced between training and test data, indicating a more generalized model.

In summary, based on the provided metrics and considering the balance between training and test performance, **Technique 3 (Data Standardization, PCA)** appears to be the best model among the four. It achieves the highest accuracy and recall on the test data while avoiding signs of overfitting.

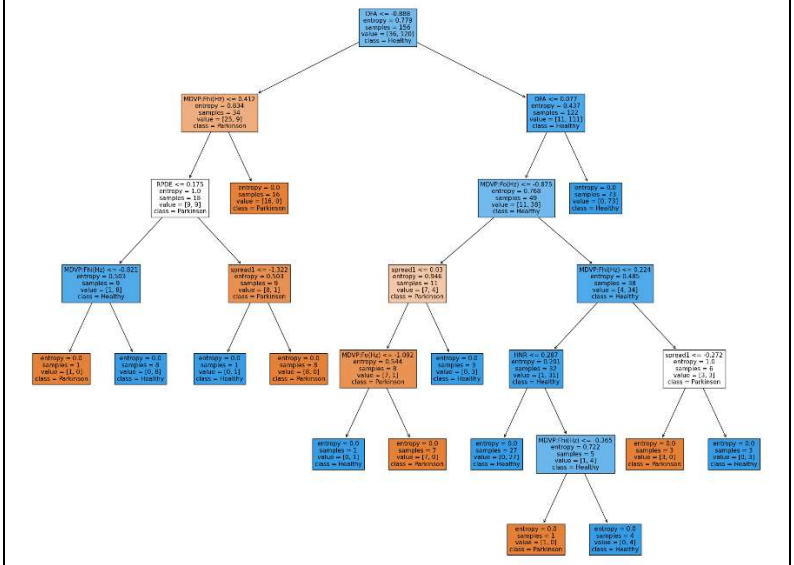
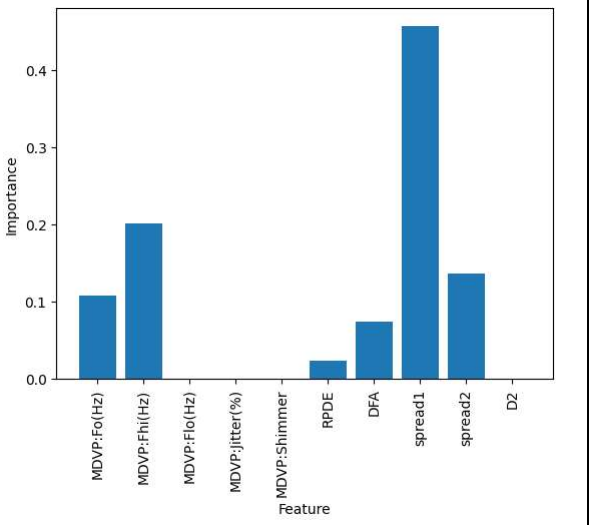
Decision Tree

A Decision Tree is a supervised machine learning algorithm that is used for both classification and regression tasks. It is a graphical representation of decisions and their consequences in the form of a tree-like structure. Decision Trees are widely used because they are easy to understand, interpret, and visualize.

Decision Trees tend to overfit the training data, which means they can capture noise in the data and perform poorly on unseen data. Techniques like pruning, limiting the tree depth, and setting minimum samples per leaf can help prevent overfitting.

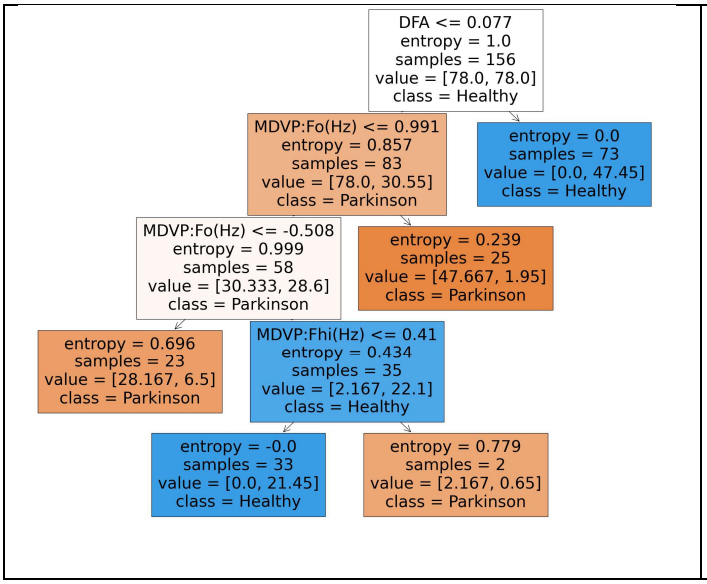
Decision Trees can provide a measure of feature importance, indicating which features are more influential in making decisions at higher levels of the tree.

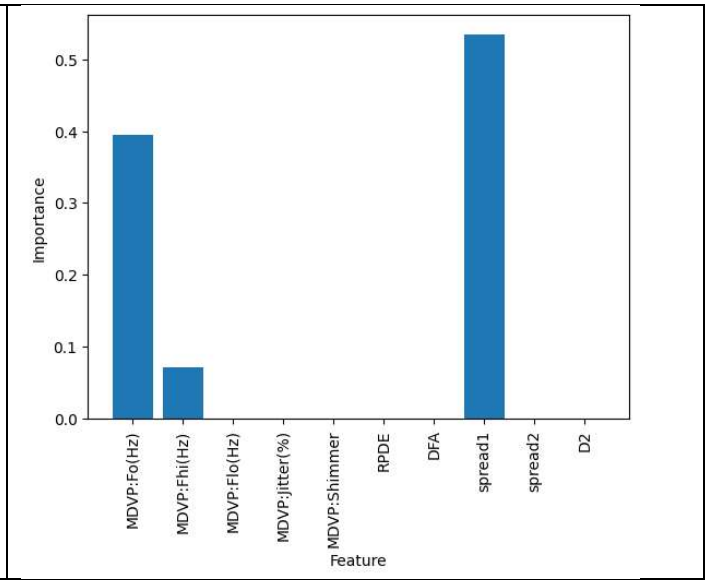
Technique 1: Feature Selection, Data Standardization

Train Data	**Test Data**
True Positive 120 False Positive 0 True Negative 36 False Negative 0 Accuracy: 1.0 Precision: 1.0 Recall: 1.0 F1_score: 1.0 confusion matrix [[36 0] [0 120]]	True Positive 25 False Positive 2 True Negative 10 False Negative 2 Accuracy: 0.8974358974358975 Precision: 0.9259259259259259 Recall: 0.9259259259259259 F1_score: 0.9259259259259259 confusion matrix [[10 2] [2 25]]
	

Technique 2: Feature Selection, Data Standardization, Decision Tree Pruning

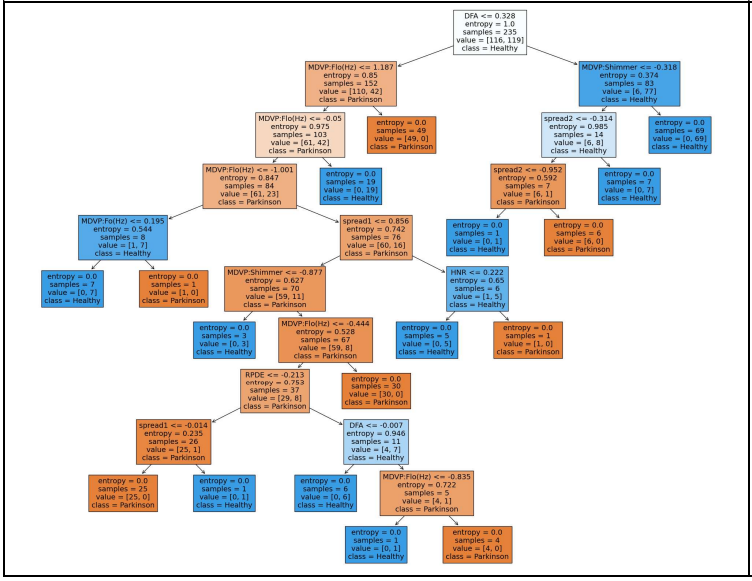
Hyperparameters:	**Train Data**	**Test Data**
random state=2 class weight='balanced' max depth=5 ccp alpha=0.05 criterion='entropy'	True Positive 106 False Positive 0 True Negative 36 False Negative 14 Accuracy: 0.9102564102564102 Precision: 1.0 Recall: 0.8833333333333333 F1_score: 0.9380530973451328 confusion matrix [[36 0] [14 106]]	True Positive 22 False Positive 2 True Negative 10 False Negative 5 Accuracy: 0.8205128205128205 Precision: 0.9166666666666666 Recall: 0.8148148148148148 F1_score: 0.8627450980392156 confusion matrix [[10 2] [5 22]]

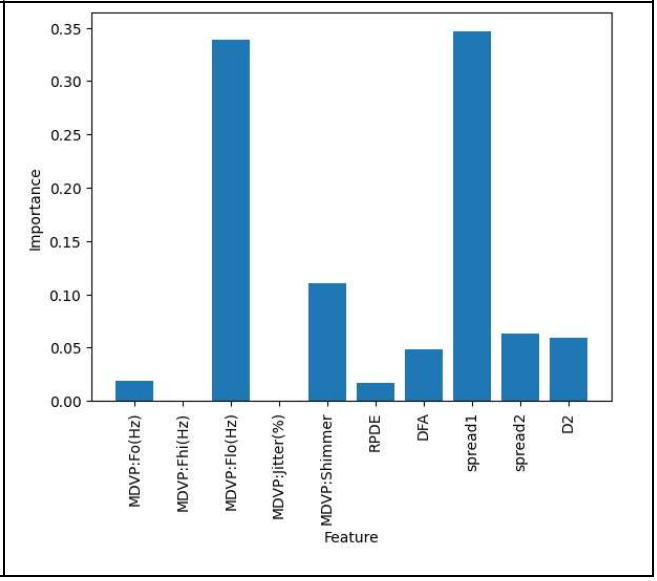




Technique 3: Feature Selection, Data Standardization, Oversampling

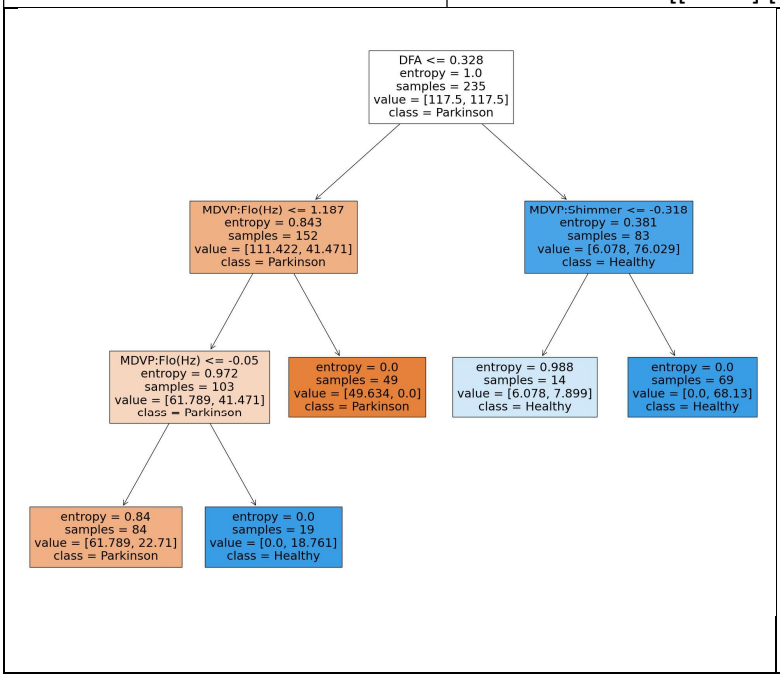
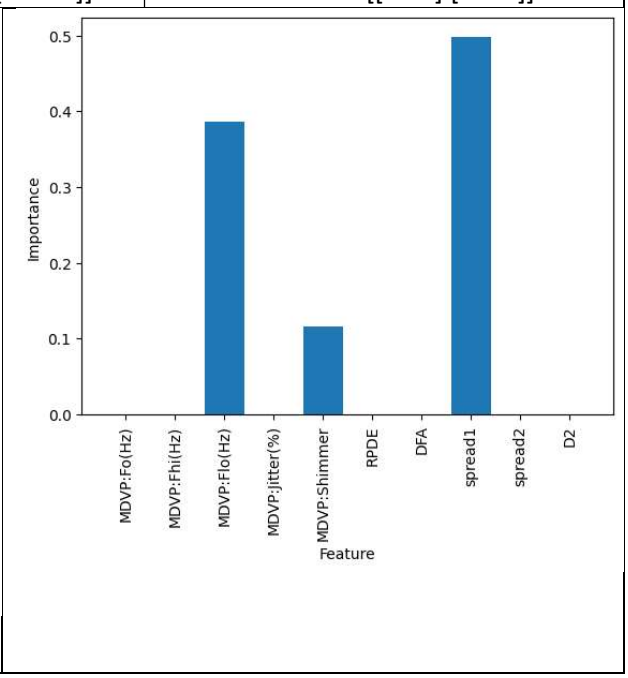
Train Data	**Test Data**
True Positive 119	True Positive 24
False Positive 0	False Positive 4
True Negative 116	True Negative 27
False Negative 0	False Negative 4
Accuracy: 1.0	Accuracy: 0.864406779661017
Precision: 1.0	Precision: 0.8571428571428571
Recall: 1.0	Recall: 0.8571428571428571
F1_score: 1.0	F1_score: 0.8571428571428571
confusion matrix [[116 0] [0 119]]	confusion matrix [[27 4] [4 24]]



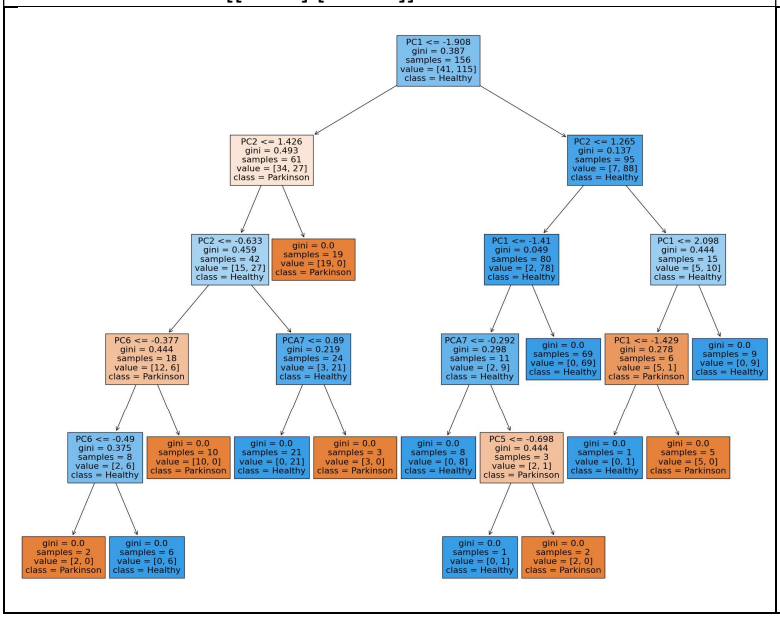
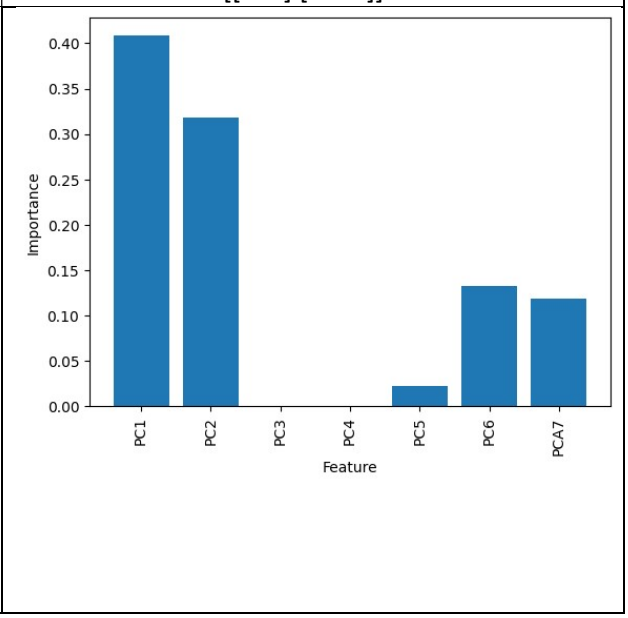


Technique 4: Feature Selection, Data Standardization, Oversampling, Decision Tree Pruning

Hyperparameters:	**Train Data**	**Test Data**
random state=2	True Positive 96	True Positive 21
class weight='balanced'	False Positive 6	False Positive 6
max depth=5	True Negative 110	True Negative 25
ccp alpha=0.05	False Negative 23	False Negative 7
criterion='entropy'	Accuracy: 0.8765957446808511	Accuracy: 0.7796610169491526
	Precision: 0.9411764705882353	Precision: 0.7777777777777778
	Recall: 0.8067226890756303	Recall: 0.75

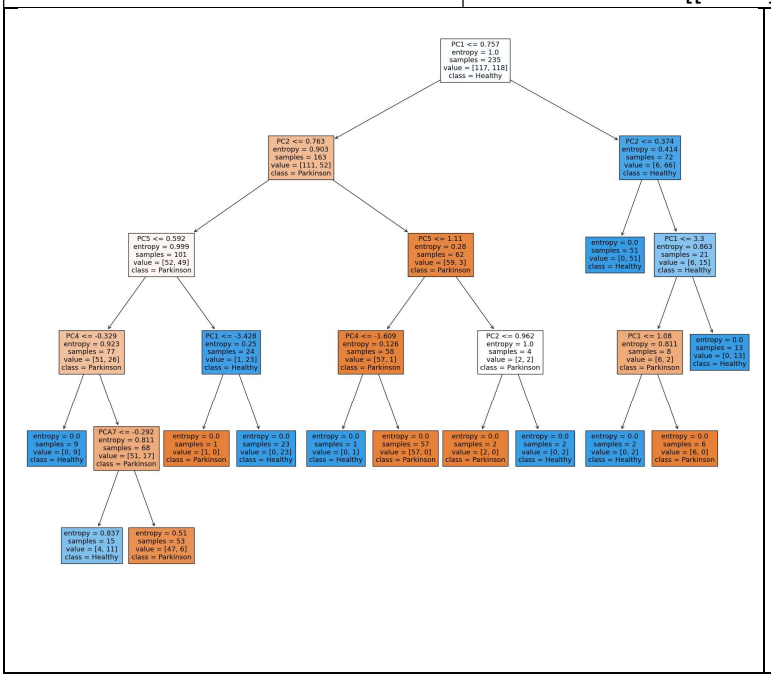
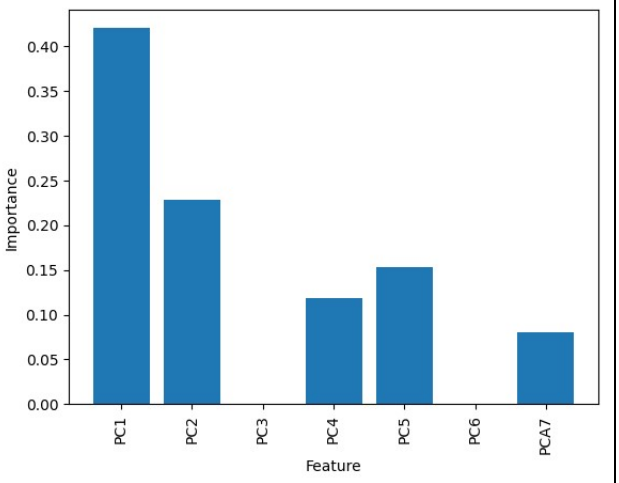
	F1_score: 0.8687782805429864 confusion matrix [[110 6] [23 96]]	F1_score: 0.7636363636363638 confusion matrix [[25 6] [7 21]]
		

Technique 5 : Data Standardization, PCA

Train Data	**Test Data**
True Positive 115 False Positive 0 True Negative 41 False Negative 0 Accuracy: 1.0 Precision: 1.0 Recall: 1.0 F1_score: 1.0 confusion matrix [[41 0] [0 115]]	True Positive 30 False Positive 2 True Negative 5 False Negative 2 Accuracy: 0.8974358974358975 Precision: 0.9375 Recall: 0.9375 F1_score: 0.9375 confusion matrix [[5 2] [2 30]]
	

Technique 6: Oversampling, Data Standardization, PCA

Hyperparameters:	**Train Data**	**Test Data**
random state=0	True Positive 112 False Positive 4	True Positive 27 False Positive 0

max depth=5	True Negative 113 False Negative 6 Accuracy: 0.9574468085106383 Precision: 0.9655172413793104 Recall: 0.9491525423728814 F1_score: 0.9572649572649573 confusion matrix [[113 4] [6 112]]	True Negative 30 False Negative 2 Accuracy: 0.9661016949152542 Precision: 1.0 Recall: 0.9310344827586207 F1_score: 0.9642857142857143 confusion matrix [[30 0] [2 27]]
		

Model Evaluation

Technique 6 (Oversampling, Data Standardization, PCA) seems to be the best model. It has the highest test accuracy and strong precision, recall, and F1_score. It also has no signs of overfitting or underfitting.

Technique 1 (Feature Selection, Data Standardization) also performs well but lacks some additional techniques used in other models to handle class imbalance and dimensionality reduction.

Technique 4 (Feature Selection, Data Standardization, Oversampling, Decision Tree Pruning) performs relatively poorly, indicating overfitting.

Techniques 2 and 3 have some potential, but they could be further improved.

The use of oversampling and PCA (Technique 6) appears to be effective in improving model performance while maintaining generalization.

Random Forest

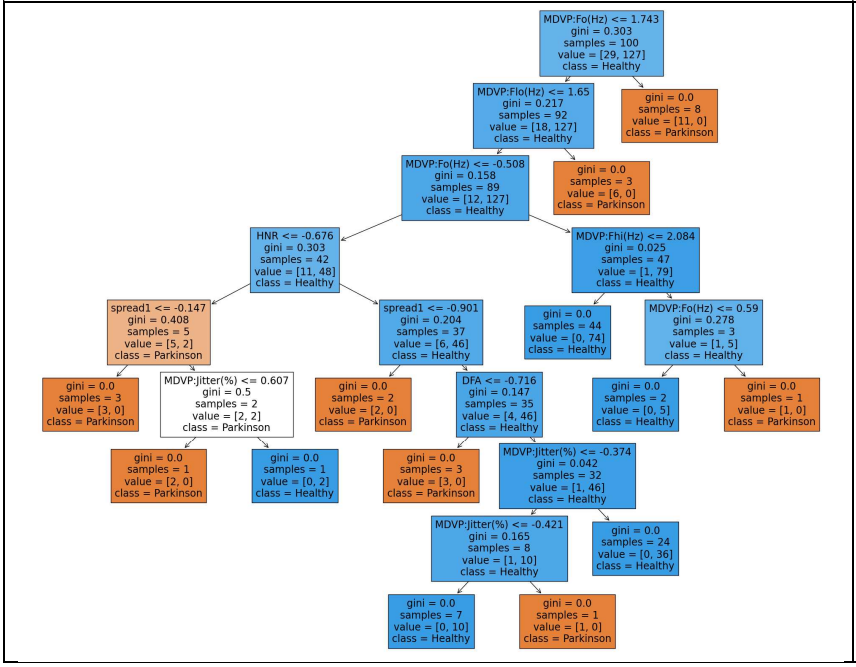
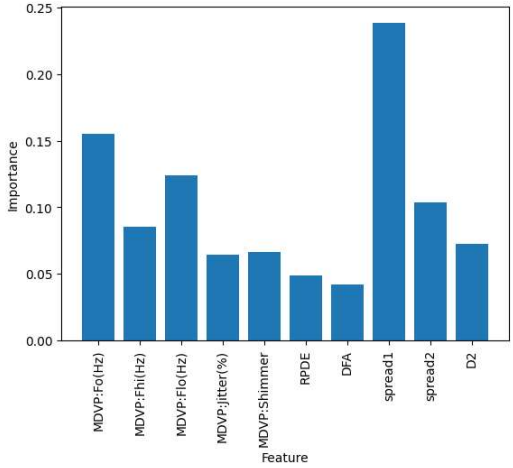
Random Forest is an ensemble learning method, which means it combines the predictions of multiple individual decision trees to make more accurate and robust predictions.

Random Forest builds multiple decision trees by randomly selecting subsets of the training data (with replacement) and training a separate tree on each subset. This reduces the risk of overfitting that can occur with a single decision tree.

In classification tasks, Random Forest typically uses a majority voting scheme, where each tree "votes" for a class, and the class with the most votes becomes the final prediction. In regression tasks, it averages the predictions of individual trees to make the final prediction.

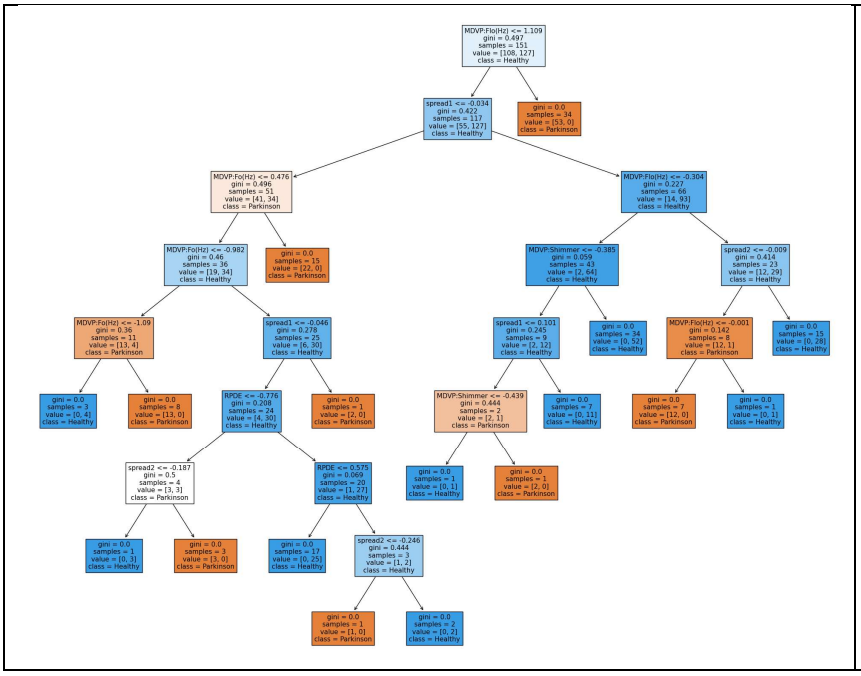
Since each tree is trained on a random subset of the data, there is a portion of the data (about 1/3) that is not used for training each tree. This OOB data can be used to estimate the model's accuracy without the need for a separate validation set.

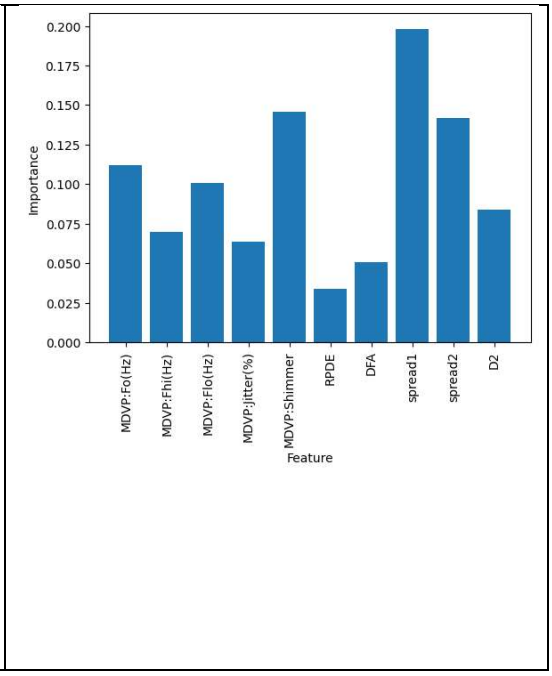
Technique 1: Feature Selection, Data Standardization

Train Data	**Test Data**
True Positive 120	True Positive 26
False Positive 0	False Positive 4
True Negative 36	True Negative 8
False Negative 0	False Negative 1
Accuracy: 1.0	Accuracy: 0.8717948717948718
Precision: 1.0	Precision: 0.8666666666666667
Recall: 1.0	Recall: 0.9629629629629
F1_score: 1.0	F1_score: 0.912280701754386
confusion matrix [[36 0] [0 120]]	confusion matrix [[8 4] [1 26]]
	

Technique 2: Feature Selection, Data Standardization, Oversampling

Train Data	**Test Data**
True Positive 119	True Positive 26
False Positive 0	False Positive 0
True Negative 116	True Negative 31
False Negative 0	False Negative 2
Accuracy: 1.0	Accuracy: 0.9661016949152542
Precision: 1.0	Precision: 1.0
Recall: 1.0	Recall: 0.9285714285714286
F1_score: 1.0	F1_score: 0.962962962962963
confusion matrix [[116 0] [0 119]]	confusion matrix [[31 0] [2 26]]

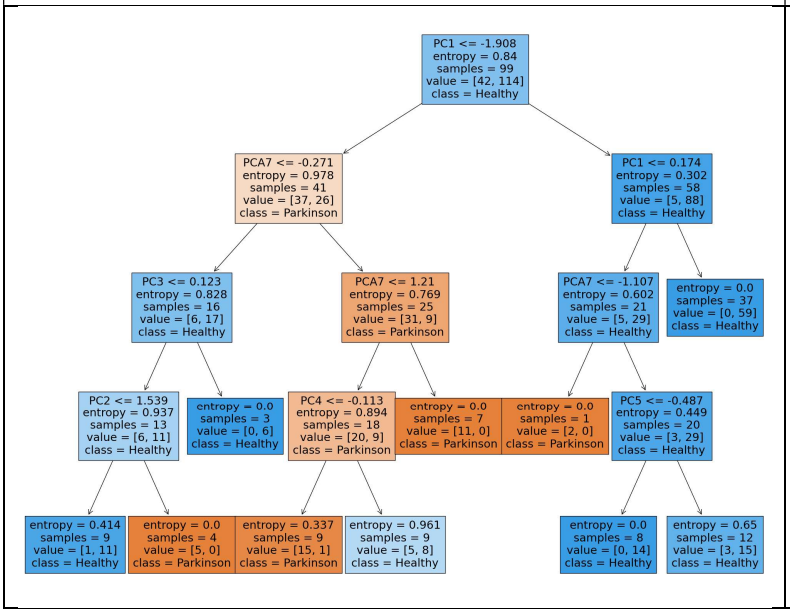


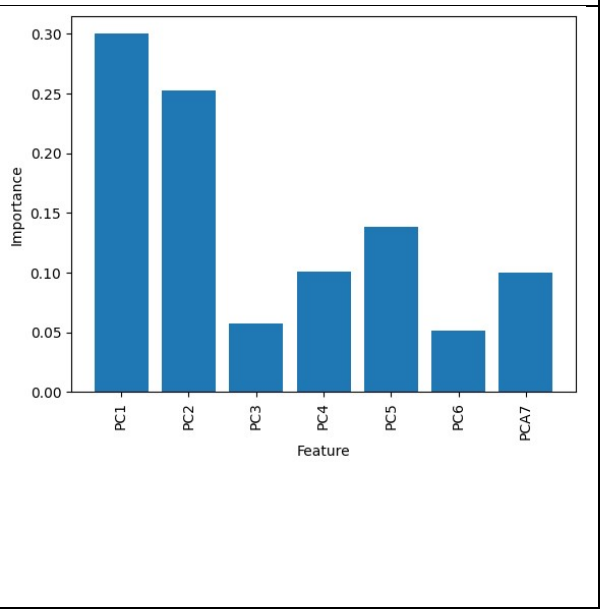


Technique 3: Data Standardization, PCA

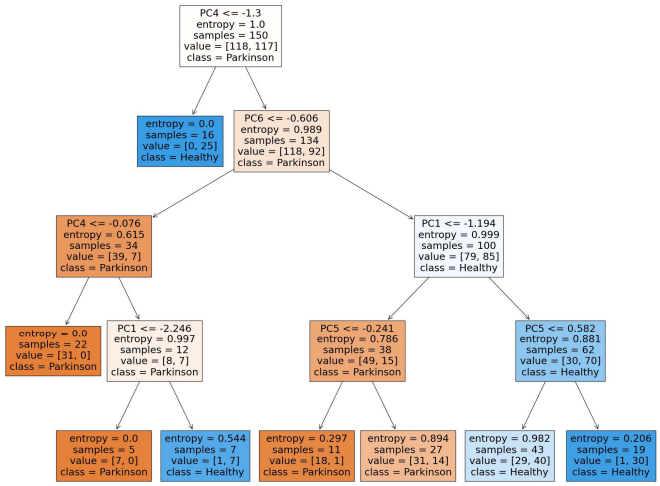
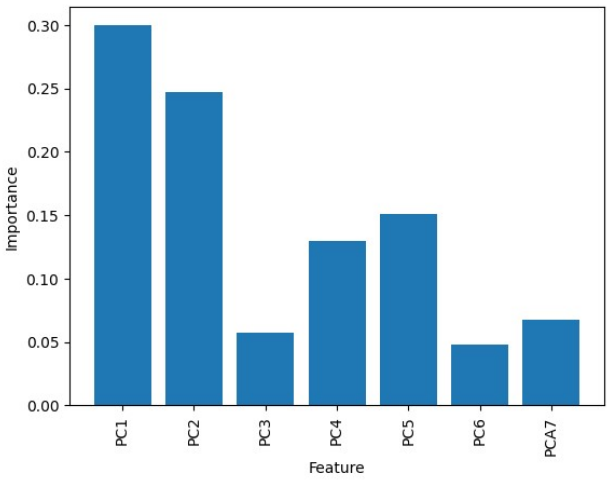
Hyperparameters
Max depth=4
criterion='entropy'
Train Data
True Positive 115
False Positive 12
True Negative 29
False Negative 0
Accuracy: 0.9230769230769231
Precision: 0.905511811023622
Recall: 1.0
F1_score: 0.9504132231404958
confusion matrix [[29 12] [0 115]]

Test Data
True Positive 30
False Positive 4
True Negative 3
False Negative 2
Accuracy: 0.8461538461538461
Precision: 0.8823529411764706
Recall: 0.9375
F1_score: 0.9090909090909091
confusion matrix [[3 4] [2 30]]





Technique 4: Data Standardization, PCA, Oversampling

Hyperparameters Max depth=4 criterion='entropy' **Train Data** True Positive 111 False Positive 1 True Negative 116 False Negative 7 Accuracy: 0.9659574468085106 Precision: 0.9910714285714286 Recall: 0.940677966101695 F1_score: 0.9652173913043479 confusion matrix [[116 1] [7 111]]	**Test Data** True Positive 29 False Positive 0 True Negative 30 False Negative 0 Accuracy: 1.0 Precision: 1.0 Recall: 1.0 F1_score: 1.0 confusion matrix [[30 0] [0 29]]
	

Model Selection

Technique 4 (Data Standardization, PCA, Oversampling) seems to be the best model, achieving a perfect test accuracy and high precision, recall, and F1_score. It shows no signs of overfitting or underfitting and performs exceptionally well.

Technique 2 (Feature Selection, Data Standardization, Oversampling) also performs well with a high-test accuracy and balanced precision and recall.

Technique 1 (Feature Selection, Data Standardization) performs reasonably well but has overfitted.

Technique 3 (Data Standardization, PCA) underperforms compared to the others, suggesting that PCA might not be the best choice for this dataset.

All techniques except technique 1 appear to handle overfitting well, as indicated by the consistency between train and test performance metrics.

XGBOOST

XGBoost, short for "Extreme Gradient Boosting," is a powerful and popular machine learning algorithm that is particularly effective for structured/tabular data and is widely used for both regression and classification tasks. It's an ensemble learning method that belongs to the gradient boosting family of algorithms.

XGBoost is an extension of gradient boosting. It builds a strong predictive model by combining the predictions of multiple weaker models (typically decision trees) sequentially. It focuses on correcting the errors made by the previous models.

XGBoost incorporates L1 (Lasso) and L2 (Ridge) regularization techniques to control overfitting. This helps prevent the model from fitting the training data noise and leads to better generalization.

Cross-validation is straightforward to implement in XGBoost, enabling robust model evaluation and hyperparameter tuning.

Technique 1: Data Standardization, Feature Selection

Train Data	**Test Data**
True Positive 120	True Positive 27
False Positive 0	False Positive 4
True Negative 36	True Negative 8
False Negative 0	False Negative 0
Accuracy: 1.0	Accuracy: 0.8974358974358975
Precision: 1.0	Precision: 0.8709677419354839
Recall: 1.0	Recall: 1.0
F1_score: 1.0	F1_score: 0.9310344827586207
confusion matrix [[36 0] [0 120]]	confusion matrix [[8 4] [0 27]]

Technique 2: Data Standardization, Feature Selection, Oversampling

Train Data	**Test Data**
True Positive 119	True Positive 27
False Positive 0	False Positive 0
True Negative 116	True Negative 31
False Negative 0	False Negative 1
Accuracy: 1.0	Accuracy: 0.9830508474576272
Precision: 1.0	Precision: 1.0
Recall: 1.0	Recall: 0.9642857142857143
F1_score: 1.0	F1_score: 0.9818181818181818
confusion matrix [[116 0] [0 119]]	confusion matrix [[31 0] [1 27]]

Technique 3: Data Standardization, PCA

Train Data	**Test Data**
True Positive 115	True Positive 29
False Positive 0	False Positive 3
True Negative 41	True Negative 4
False Negative 0	False Negative 3
Accuracy: 1.0	Accuracy: 0.8461538461538461
Precision: 1.0	Precision: 0.90625
Recall: 1.0	Recall: 0.90625
F1_score: 1.0	F1_score: 0.90625
confusion matrix [[41 0] [0 115]]	confusion matrix [[4 3] [3 29]]

Technique 4: Data Standardization, PCA, Oversampling

Hyperparameters	**Test Data**
max depth=4 random state=2 n estimators=100 **Train Data** True Positive 118 False Positive 0 True Negative 117 False Negative 0 Accuracy: 1.0 Precision: 1.0 Recall: 1.0 F1_score: 1.0 confusion matrix [[117 0] [0 118]]	True Positive 29 False Positive 0 True Negative 30 False Negative 0 Accuracy: 1.0 Precision: 1.0 Recall: 1.0 F1_score: 1.0 confusion matrix [[30 0] [0 29]]

Model Evaluation

Technique 4 (Data Standardization, PCA, Oversampling) appears to be the best model. It achieves a perfect test accuracy and high precision, recall, and F1_score. It shows no signs of overfitting or underfitting and performs exceptionally well.

Technique 2 (Data Standardization, Feature Selection, Oversampling) also performs very well with a high test accuracy and balanced precision and recall.

Technique 1 (Data Standardization, Feature Selection) performs reasonably well but has room for improvement, especially compared to Techniques 2 and 4.

Technique 3 (Data Standardization, PCA) underperforms compared to the others, suggesting that PCA might not be the best choice for this dataset.

All techniques appear to handle overfitting well, as indicated by the consistency between train and test performance metrics.

In this case, Technique 4 is the best model, as it achieves the highest test accuracy and perfectly balances precision and recall.

Anomaly of Decision Trees and Ensemble models with continuous data

Overfitting:

- Decision trees, especially deep ones, can be prone to overfitting. This means they may capture noise or outliers in the training data as if they were genuine patterns. As a result, the model might perform poorly on new, unseen data.
- Ensemble models, such as Random Forests or Gradient Boosted Trees, are designed to mitigate overfitting by combining the predictions of multiple trees. However, if individual trees in the ensemble overfit to anomalies, the ensemble might still be affected.

Outliers:

- Decision trees can be sensitive to outliers, as they might create splits specifically to accommodate these extreme values. This behavior could lead to anomalies where the model heavily relies on unusual data points.
- Some ensemble models, like Isolation Forests, are specifically designed to handle outliers. However, traditional ensembles like Random Forests might struggle with outliers if individual trees are sensitive to them.

Resolution Sensitivity:

- The resolution of the continuous data might affect the model's sensitivity to anomalies. For example, if a feature has a high resolution, decision tree splits might occur at values that capture noise or outliers.

Skewed Distributions:

- Decision trees might struggle with skewed distributions, especially if the anomalies are present in the minority class. This can lead to difficulties in properly classifying the anomalies.

These are some of the reasons why our above models are overfitting and we are most likely going to discard them.

Support Vector Machine (SVM)

Support Vector Machine (SVM) is a powerful supervised machine learning algorithm that is primarily used for classification tasks, although it can also be extended to regression tasks. SVMs are known for their ability to handle both linear and non-linear classification problems effectively.

SVM can perform linear classification by finding a hyperplane that best separates two classes of data points. It can also handle non-linear classification by using kernel tricks to map the data into a higher-dimensional space where a linear separator can be found.

SVM aims to find a hyperplane that maximizes the margin (the distance between the hyperplane and the nearest data points of each class). This results in a decision boundary that is less sensitive to noise and leads to better generalization.

Kernel functions are used to implicitly map the data into a higher-dimensional space. Common kernel functions include Linear, Polynomial, Radial Basis Function (RBF), and Sigmoid. The choice of the kernel function depends on the problem and data.

SVM has a regularization parameter (C) that controls the trade-off between maximizing the margin and minimizing classification errors. A smaller C encourages a larger margin but allows some misclassifications, while a larger C reduces the margin but minimizes misclassifications.

SVM is robust to outliers because it focuses on the support vectors that are typically not influenced by outliers. SVM can perform well even when the number of features (dimensions) is much larger than the number of data points.

SVM works well with continuous data, and sometimes better than ANN, as it uses less computational resources.

Hyperparameter : kernel="linear"

Technique 1: Data Standardization, Feature Selection

Train Data	**Test Data**
True Positive 119	True Positive 27
False Positive 15	False Positive 5
True Negative 21	True Negative 7
False Negative 1	False Negative 0
Accuracy: 0.8974358974358975	Accuracy: 0.8717948717948718
Precision: 0.8880597014925373	Precision: 0.84375
Recall: 0.9916666666666667	Recall: 1.0
F1_score: 0.937007874015748	F1_score: 0.9152542372881356
confusion matrix [[21 15] [1 119]]	confusion matrix [[7 5] [0 27]]

Technique 2: Data Standardization, Feature Selection, Oversampling

Train Data	**Test Data**
True Positive 101	True Positive 24
False Positive 12	False Positive 2
True Negative 104	True Negative 29
False Negative 18	False Negative 4

Accuracy: 0.8723404255319149 Precision: 0.8938053097345132 Recall: 0.8487394957983193 F1_score: 0.8706896551724138 confusion matrix [[104 12] [18 101]]	Accuracy: 0.8983050847457628 Precision: 0.9230769230769231 Recall: 0.8571428571428571 F1_score: 0.888888888888889 confusion matrix [[29 2] [4 24]]

Technique 3: Data Standardization, PCA

Train Data	**Test Data**
True Positive 111 False Positive 18 True Negative 23 False Negative 4 Accuracy: 0.8589743589743589 Precision: 0.8604651162790697 Recall: 0.9652173913043478 F1_score: 0.9098360655737704 confusion matrix [[23 18] [4 111]]	True Positive 32 False Positive 4 True Negative 3 False Negative 0 Accuracy: 0.8974358974358975 Precision: 0.8888888888888888 Recall: 1.0 F1_score: 0.9411764705882353 confusion matrix [[3 4] [0 32]]

Technique 4: Data Standardization, PCA, Oversampling

Train Data	**Test Data**
True Positive 94 False Positive 9 True Negative 108 False Negative 24 Accuracy: 0.8595744680851064 Precision: 0.912621359223301 Recall: 0.7966101694915254 F1_score: 0.8506787330316742 confusion matrix [[108 9] [24 94]]	True Positive 25 False Positive 1 True Negative 29 False Negative 4 Accuracy: 0.9152542372881356 Precision: 0.9615384615384616 Recall: 0.8620689655172413 F1_score: 0.9090909090909091 confusion matrix [[29 1] [4 25]]

Model Evaluation

Technique 1 (Data Standardization, Feature Selection) appears to be the best model. It achieves a high test accuracy and balances precision and recall well. There is no apparent overfitting or underfitting.

Technique 3 (Data Standardization, PCA) also performs well with a good balance between precision and recall.

Technique 2 (Data Standardization, Feature Selection, Oversampling) shows an improvement over Technique 1 but still has some imbalance between precision and recall.

Technique 4 (Data Standardization, PCA, Oversampling) appears to overfit the training data, as indicated by the lower test accuracy and the imbalance between precision and recall.

In this case, Technique 4 is the best model, as it achieves a high test accuracy and well-balanced precision and recall.

KNN Model

K-Nearest Neighbors (KNN) is a simple and intuitive machine learning algorithm used for classification and regression tasks. The fundamental idea behind KNN is to predict the label of a data point by considering the labels of its nearest neighbors in the feature space.

KNN relies on a distance metric (e.g., Euclidean distance, Manhattan distance) to measure the similarity between data points in the feature space.

KNN is sensitive to the scale of features, so it's often beneficial to scale or normalize the features before applying the algorithm.

KNN can be computationally expensive, especially as the size of the dataset increases. Efficient data structures like KD-trees or Ball trees can be used to speed up the search for nearest neighbors.

- No training phase (lazy learning)
- Suitable only for small datasets
- Sensitive to irrelevant or redundant features

Technique 1: Data Standardization, Feature Selection

Train Data	**Test Data**
True Positive 118	True Positive 26
False Positive 4	False Positive 3
True Negative 32	True Negative 9
False Negative 2	False Negative 1
Accuracy: 0.9615384615384616	Accuracy: 0.8974358974358975
Precision: 0.9672131147540983	Precision: 0.896551724137931
Recall: 0.9833333333333333	Recall: 0.9629629629629629
F1_score: 0.9752066115702478	F1_score: 0.9285714285714286
confusion matrix	confusion matrix
[[32 4]	[[9 3]
[2 118]]	[1 26]]

Technique 2: Data Standardization, Feature Selection, Oversampling

Train Data	**Test Data**
True Positive 109	True Positive 23
False Positive 0	False Positive 0
True Negative 116	True Negative 31
False Negative 10	False Negative 5
Accuracy: 0.9574468085106383	Accuracy: 0.9152542372881356
Precision: 1.0	Precision: 1.0
Recall: 0.9159663865546218	Recall: 0.8214285714285714
F1_score: 0.9561403508771931	F1_score: 0.9019607843137255
confusion matrix	confusion matrix
[[116 0]	[[31 0]
[10 109]]	[5 23]]

Technique 3: Data Standardization, PCA

Train Data	**Test Data**
True Positive 104	True Positive 32
False Positive 2	False Positive 2
True Negative 39	True Negative 5
False Negative 11	False Negative 0
Accuracy: 0.9166666666666666	Accuracy: 0.9487179487179487
Precision: 0.9811320754716981	Precision: 0.9411764705882353
Recall: 0.9043478260869565	Recall: 1.0
F1_score: 0.9411764705882354	F1_score: 0.9696969696969697
confusion matrix	confusion matrix
[[39 2]	[[5 2]
[11 104]]	[0 32]]

Technique 4: Data Standardization, PCA, Oversampling

Train Data	**Test Data**
True Positive 97	True Positive 28
False Positive 1	False Positive 1
True Negative 116	True Negative 29

False Negative 21 Accuracy: 0.9063829787234042 Precision: 0.9897959183673469 Recall: 0.8220338983050848 F1_score: 0.898148148148148 confusion matrix [[116 1] [21 97]]	False Negative 1 Accuracy: 0.9661016949152542 Precision: 0.9655172413793104 Recall: 0.9655172413793104 F1_score: 0.9655172413793104 confusion matrix [[29 1] [1 28]]

Model Evaluation

Technique 4 (Data Standardization, PCA, Oversampling) performs well on both training and test data, suggesting good generalization without overfitting.

ANN Model

Creating an artificial neural network (ANN) involves defining the architecture of the network, including the number of layers, the number of neurons in each layer, the activation functions, and the learning parameters.

Dropout is a regularization technique commonly used in artificial neural networks (ANNs) to prevent overfitting. Overfitting occurs when a model performs well on the training data but fails to generalize to new, unseen data. Dropout helps mitigate overfitting by randomly dropping (i.e., setting to zero) a subset of units (neurons) during training.

Model Layers

```
model.add(Dense(10, input_dim=10, activation='relu'))

model.add(Dropout(0.2))

model.add(Dense(200, activation='relu'))

model.add(Dropout(0.2))

model.add(Dense(200, activation='relu'))

model.add(Dropout(0.4))

model.add(Dense(200, activation='relu'))

model.add(Dense(1, activation='sigmoid')) #output layer
```

Model Loss function : binary cross entropy

Technique 1: Data Standardization, Feature Selection

Train Data	**Test Data**
loss: 0.0952 accuracy: 0.9679	loss: 0.2882 accuracy: 0.9231

Technique 2: Data Standardization, Feature Selection, Oversampling

Train Data	**Test Data**
loss: 0.0695 accuracy: 0.9787	loss: 0.1159 accuracy: 0.9661

Technique 3: Data Standardization, PCA

Train Data	**Test Data**
loss: 0.1293	loss: 0.3304

accuracy: 0.9551	accuracy: 0.8718

Technique 4: Data Standardization, PCA, Oversampling

Train Data	**Test Data**
loss: 0.1101 accuracy: 0.9745	loss: 0.0907 accuracy: 0.9831

Model Evaluation

Technique 2 (Data Standardization, Feature Selection, Oversampling) and Technique 4 (Data Standardization, PCA, Oversampling) show good generalization with minimal differences between training and test accuracy.

Possible Overfitting: Technique 3 (Data Standardization, PCA) shows a significant drop in accuracy on the test data compared to the training data, indicating potential overfitting.

Overall Best Model

The best models working for this problem statement are SVM, KNN and ANN model with dropout, with ANN performing the best.

Best ANN Model: Feature Selection, Data Standardization, and SMOTE Oversampling

For our best performing artificial neural network (ANN) model, we employed a meticulous approach to enhance both feature quality and data balance, leading to superior predictive performance. The key steps involved in crafting this robust model are detailed below:

1. **Feature Selection:**
    - We prioritized the quality of input features by conducting feature selection, specifically removing highly correlated data. This process ensures that the model focuses on the most informative aspects of the dataset, contributing to improved generalization.
2. **Data Standardization:**
    - To facilitate consistent and meaningful comparisons between features, we applied data standardization. This step ensures that all features are on a similar scale, preventing any single feature from dominating the learning process. Standardization contributes to a stable and efficient training process.
3. **SMOTE Oversampling:**
    - Addressing class imbalance is crucial for training a model that is sensitive to all classes. We leveraged the Synthetic Minority Over-sampling Technique (SMOTE) to balance the class distribution. This approach involves generating synthetic samples of the minority class, creating a more representative dataset and preventing the model from being biased towards the majority class.
4. **Model Training Configuration:**
    - For training the ANN, we chose Binary Cross Entropy as the loss function. This loss function is suitable for binary classification tasks, aligning with the nature of our problem.
    - We employed the Adam optimizer, a widely used optimization algorithm known for its efficiency and effectiveness in optimizing neural networks.
5. **Evaluation Metric:**
    - To measure the model's overall performance, we utilized accuracy as the evaluation metric. Accuracy provides a comprehensive view of the model's ability to correctly classify instances, making it a meaningful metric for our binary classification task.

This meticulous combination of feature selection, data standardization, SMOTE oversampling, and thoughtful model training parameters resulted in an ANN that excels in both training and generalization. The model demonstrates robustness, effectively addressing challenges such as class imbalance and high feature correlation. Its performance is summarized using accuracy, a metric that aligns with our goal of achieving accurate and balanced predictions.

This approach not only enhances the model's predictive power but also ensures its reliability in real-world scenarios where class imbalances and correlated features are common challenges.