Internship Report

October 2024

National Centre for Physics

Submitted By:

Syed Atiq Ali Shah
Muhammad Yahya Khan

Submitted To:

Dr. Ashfaq Ahmad

Experimental High Energy Physics Department (EHEP)

Executive Summary:

During my internship at the National Center for Physics (NCP), I applied machine learning techniques using the ROOT framework to analyze high-energy physics data. Initially, I explored machine learning fundamentals and various supervised algorithms, including regression and classification models. The main focus of my work was to distinguish signal events from background events using ROOT’s TMVA library. Out of 21 variables in the dataset, I identified five key variables ('numJets', 'Scalar_HT', 'jet6_pt', 'jet5_pt', 'jet4_pt') as most prominent using a random forest model for feature importance. To verify this selection, I trained a boosted decision tree on both the complete dataset and the reduced set containing only these top variables, then compared their efficiencies. The results showed a 0.02 difference in model efficiency, indicating that training on just the top variables yields nearly identical accuracy. This analysis identified critical features that effectively distinguish Four Top-W boson signal events from background data.

Table of Contents

Table of Figures

1 Introduction:

1.1 Background:

High-energy physics, particularly the study of particle collisions, is an area of research focused on understanding the fundamental building blocks of the universe. This field often involves experiments using particle accelerators like the Large Hadron Collider (LHC) at CERN, where protons are collided at nearly the speed of light, generating a variety of subatomic particles. These collisions produce complex datasets that record the properties and behaviors of the resulting particles, making data analysis a critical part of particle physics research.

Machine Learning (ML) has emerged as a powerful tool for analyzing such vast and intricate datasets. ML techniques help physicists uncover patterns in data, make predictions, and distinguish between different types of particle events, which is crucial for identifying rare occurrences like the decay of a Four Top-W boson among a multitude of background events. ML automates the extraction of insights from complex data, significantly reducing the time and effort needed to interpret results from large-scale experiments.

Machine learning is broadly categorized into three types:

- **Supervised Learning**: It involves training algorithms on labeled datasets, allowing the model to learn the relationship between input variables and their corresponding output. Supervised learning is particularly useful in classifying events in particle physics, where each collision can be tagged as a specific type of event.

- **Unsupervised Learning**: This method deals with data without labels, finding patterns or clusters in data. In particle physics, it can help discover new phenomena by grouping data points that have similar properties.

- **Reinforcement Learning**: This type involves agents that learn to make decisions by interacting with their environment and maximizing cumulative rewards. Though less common in particle event classification, it has applications in optimizing data processing tasks.

Among these, **Supervised Learning** plays a key role in particle physics for classifying and predicting outcomes based on labeled collision data. By training on datasets where particle events are categorized, algorithms can learn to distinguish **signal events**—which indicate the presence of a specific particle decay, like Four Top-W bosons—from **background events**, which represent all other non-signal occurrences.

A popular tool for high-energy physics analysis is **ROOT**, a software framework developed at CERN. ROOT provides a comprehensive suite for data analysis, visualization, and storage. It is widely used in particle physics for its ability to handle large datasets and perform detailed statistical analysis. ROOT includes various libraries like **TH1F** for 1D histograms and **TH2F** for 2D histograms, which are essential for visualizing distributions of particle events. The **TMVA (Toolkit for Multi-Variate Analysis)** library within ROOT integrates machine learning algorithms, making it possible to apply classification techniques directly to high- energy physics data.

The analysis of particle collision data often involves distinguishing rare **signal events** from a background of more common interactions. For example, when protons are collided, the interactions produce a range of particles that decay into further particles. Among these, the decay of a **Four Top-W boson** is a particularly rare event. Detecting this requires analyzing millions of collision events to identify the unique characteristics that set these signal events apart from other background events.

In this internship project, the focus was on analyzing ROOT data containing a **SignalTree** and a **BackgroundTree**, representing Four Top-W boson events and other background phenomena, respectively. By converting the ROOT data into a more accessible format (Excel) and comparing variables from these two datasets, the project aimed to identify the ranges of variable values where the signal events occurred distinctly from the background events. Visual tools such as histograms, density plots, and boxplots were employed to illustrate these differences, facilitating the identification of regions where signal events were more likely to be observed.

This background provides the context for the internship's objective: using supervised machine learning to classify particle events and leveraging the ROOT framework for effective data analysis in high-energy physics. The integration of ML with tools like ROOT not only enhances our ability to process complex datasets but also accelerates the discovery of new physics phenomena.

1.2 Problem Statement:

High-energy physics experiments, like those at the LHC, generate complex datasets where rare Four Top-W boson decays are hidden among many background events. This project aims to:

Use supervised machine learning to distinguish these rare signal events from background noise.

Leverage the ROOT framework for effective analysis and visualization.

The objective is to enhance accuracy in event classification, supporting advanced particle physics research.

1.3 Objectives:

The main objective of this report is to develop and apply supervised machine learning techniques, integrated with the ROOT framework, to accurately distinguish Four Top-W boson decay events from background events in high- energy physics data, thereby improving the efficiency and precision of data analysis in particle physics research.

1.4 Scope:

The scope of this report includes an overview of machine learning, focusing on supervised, unsupervised, and reinforcement learning, with examples relevant to particle physics. It provides a detailed explanation of key supervised learning algorithms, such as regression methods, decision trees, ensemble methods, support vector machines, and neural networks. The report also introduces the ROOT framework, highlighting essential libraries like TH1F, TH2F, and TMVA, with practical examples for easy understanding. It then applies these machine learning techniques to analyze ROOT data, specifically differentiating Four Top- W boson events from background events using data visualization methods. Finally, the report offers practical guidance for using these techniques in high- energy physics data analysis, making it a useful resource for researchers and students.

1.5 Limitations:

The limitations of this report include a focus on basic machine learning algorithms, which may overlook advanced techniques. The analysis is confined to a specific dataset of Four Top-W boson decays, limiting its generalization. It assumes a basic understanding of ROOT and machine learning, which may hinder accessibility for beginners. Additionally, due to time constraints, the mathematical details of algorithms are only briefly covered.

2 Supervised Machine Learning Algorithms:

Supervised machine learning is a type of machine learning where the model is trained on a labeled dataset, meaning that each training example is paired with an output label. The goal is to learn a mapping from inputs to outputs so that the model can make accurate predictions on new, unseen data.

There are two main types of supervised machine learning:

1. Classification: This involves predicting a discrete label or category. For example, classifying emails as "spam" or "not spam" based on their content. The model learns from labeled examples and assigns a category to new instances based on the patterns it has identified.

2. Regression: In this case, the objective is to predict a continuous value. For instance, predicting house prices based on features like size, location, and number of bedrooms. The model learns to map input features to a numerical output.

Algorithms are the specific methods or techniques used to perform the learning task in supervised machine learning. They define how the model will learn from the training data and make predictions. Common algorithms include:

2.1 Linear Regression:

Linear regression is a supervised learning algorithm used to model the relationship between one or more independent variables (features) and a continuous dependent variable (target). The main idea is to fit a linear equation to the observed data.

Mathematical Form:

The linear regression model can be expressed mathematically as:

$$y=\beta_0+\beta_1 x+\epsilon$$

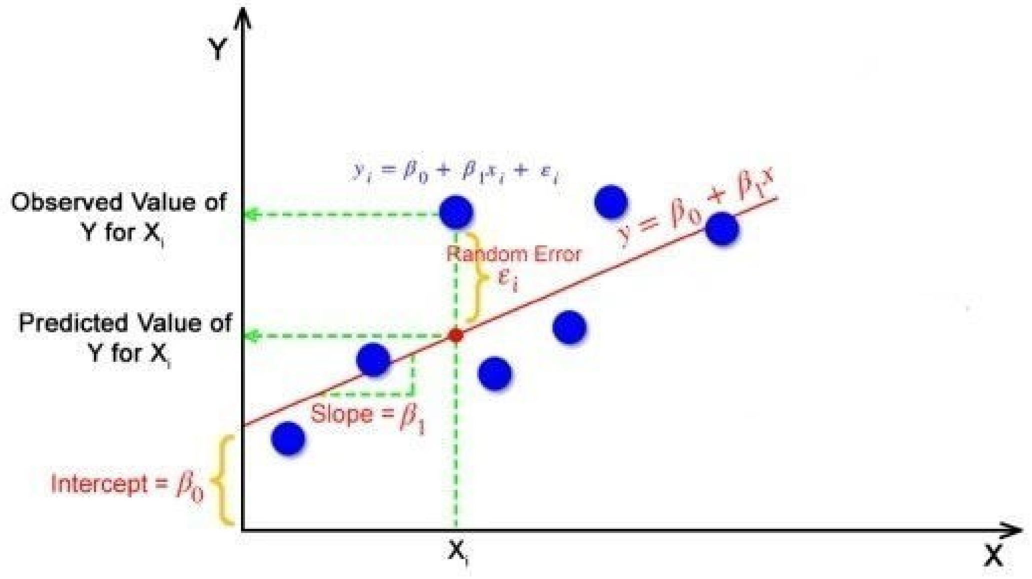

Y
$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$
Observed Value of Y for X_i
$y = \beta_0 + \beta_1 x$
Random Error ε_i
Predicted Value of Y for X_i
Slope = β_1
Intercept = β_0
X_i
X

Figure 1: Linear Regression

- y is the dependent variable (the value we want to predict).
- x is the independent variables (features).
- β_0 is the intercept (the value of y when all x values are 0).
- β_1 is slope.
- ϵ is the error term, accounting for the variability in y.

Uses:

Linear regression is widely used in various fields for:

- **Predictive Analysis:** Estimating future values based on historical data, such as predicting sales or housing prices.
- **Trend Analysis**: Identifying trends and relationships between variables, such as the impact of advertising on sales.
- **Statistical Inference**: Assessing the significance of predictors and understanding the relationships in data.

Limitations:

- **Linearity Assumption**: Assumes a linear relationship between independent and dependent variables, which may not hold true for all datasets.
- **Sensitivity to Outliers**: Outliers can disproportionately influence the results, leading to inaccurate predictions.
- **Multicollinearity**: High correlation among independent variables can lead to instability in the coefficient estimates.
- **Limited Expressiveness**: May not capture complex relationships in data, requiring more advanced models for non-linear patterns.

2.2 Multiple Linear Regression:

Multiple Linear Regression:2.2 Multiple Linear Regression:Multiple linear regression is an extension of simple linear regression that models the relationship between two or more independent variables and a single dependent variable. It aims to understand how the dependent variable changes as a function of multiple independent variables.

Mathematical Form:

The mathematical representation of multiple linear regression is:

$$y=\beta_0+\beta_1x_1+\beta_2x_2+\ldots+\beta_nx_n+\epsilon$$

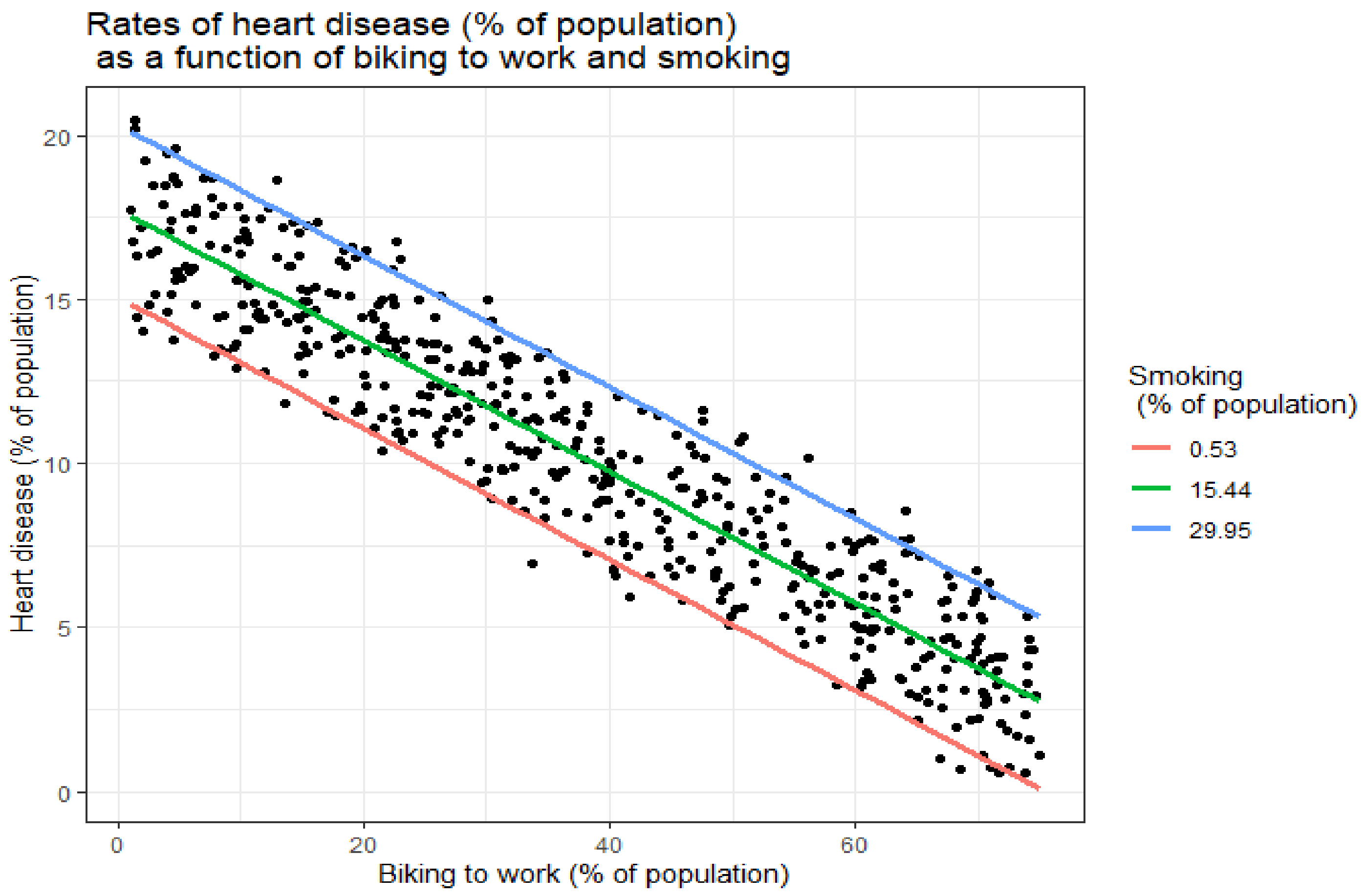


Figure 2: Multiple Linear Regression

- y is the dependent variable (the outcome we want to predict).
- $x_1, x_2, \ldots, x_n$ are the independent variables (features).
- β_0 is the intercept (the predicted value of y when all x values are zero).
- $\beta_1, \beta_2, \ldots, \beta_n$ are the coefficients representing the change in y for a one-unit change in the respective x variable, assuming all other variables remain constant.
- ϵ is the error term, capturing the variation in y.

Uses:

Multiple linear regression is widely used for various purposes, including:

- **Predictive Modeling**: Estimating outcomes based on several predictors, such as predicting house prices based on features like size, location, and number of rooms.

- **Quantifying Relationships**: Determining how multiple factors collectively affect a dependent variable, such as the impact of various economic indicators on GDP.

- **Evaluating Interactions**: Assessing whether the effect of one independent variable on the dependent variable changes depending on the value of another independent variable.

Limitations:

- **Linearity Assumption**: Like simple linear regression, it assumes a linear relationship between independent variables and the dependent variable, which may not always be the case

- **Multicollinearity**: High correlation between independent variables can make it difficult to determine the individual effect of each predictor and can inflate the variance of the coefficient estimates.

- **Overfitting**: Including too many independent variables can lead to a model that fits the training data well but performs poorly on unseen data.

- **Sensitivity to Outliers**: Outliers can significantly affect the regression coefficients, leading to misleading results.

2.3 Poisson Regression:

Poisson regression is a type of generalized linear model (GLM) that is used for modeling count data and contingency tables. It is particularly suited for data that follows a **Poisson distribution,** where the outcome variable represents counts (non-negative integers like 0, 1, 2, etc.). The Poisson distribution assumes that the mean and variance of the distribution are equal, making it appropriate for data where the counts are relatively low and often zero.

Key Concepts:

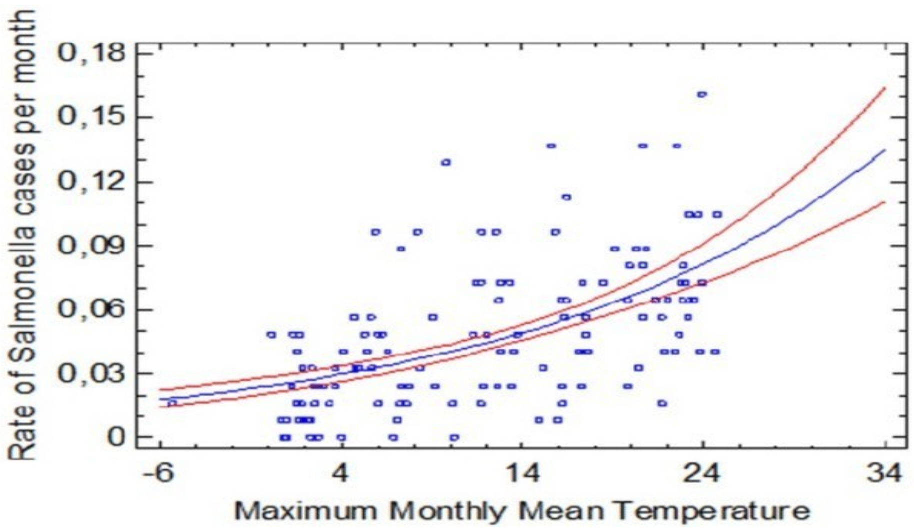
- **Response Variable (Y)**: Represents the count of occurrences of an event. For example, the number of times a specific event happens in a fixed period or area (e.g., the number of accidents per month).

- **Poisson Distribution**: The distribution assumes that the counts are random, discrete occurrences that happen independently over a fixed period of time or space.

Mathematical Form:

In Poisson regression, the mean of the count variable Y is modeled as an exponential function of a linear combination of predictors:

$$\log(\mu i)=\beta_0+\beta_1 X_1+\beta_2 X_2+\ldots+\beta_n X_n$$



Or

Figure 3: Poisson Regression

$$\mu i=\exp(\beta 0+\beta 1X1+\beta 2X2+\ldots+\beta nXn$$

- Here, μi is the expected count for observation i.
- $\beta_0,\beta_1,\ldots,\beta_n$ are the coefficients that describe the relationship between the predictors ($X_1,X_2,\ldots,X_n$) and the log of the expected count.

Characteristics:

- **Link Function**: The log link function log(μ) is used to ensure that the predicted counts remain positive.
- **Assumptions**:
    - The count data Y follows a Poisson distribution.
    - The counts are independent of each other.
    - The logarithm of the expected counts is a linear function of the explanatory variables.

Use Cases:

- Modeling the number of times a particular event occurs (e.g., the number of accidents at a location).
- Epidemiological studies, like modeling the number of disease cases.
- Insurance claims, where you want to predict the number of claims per time period.

Limitations:

- **Overdispersion**: When the variance is greater than the mean, the data may be overdispersed. In such cases, a Negative Binomial regression may be more appropriate.
- **Zero Inflation**: If the data has an excess number of zero counts, zero- inflated Poisson or hurdle models can be used for better modeling.

2.4 Polynomial Regression:

Polynomial regression is a type of regression analysis where the relationship between the independent variable X and the dependent variable Y is modeled as an n-degree polynomial. Unlike linear regression, which models a straight-line relationship between the variables, polynomial regression can capture more complex, curved patterns in the data.

Key Concepts:

- **Degree of Polynomial**: Determines the highest power of X in the model. For example, a second-degree polynomial is quadratic, a third-degree is cubic, and so on.
- **Equation Form**:
  - For a second-degree (quadratic) polynomial regression:
    $Y=\beta_0+\beta_1X+\beta_2X^2+\epsilon$
  - For a third-degree (cubic) polynomial regression:

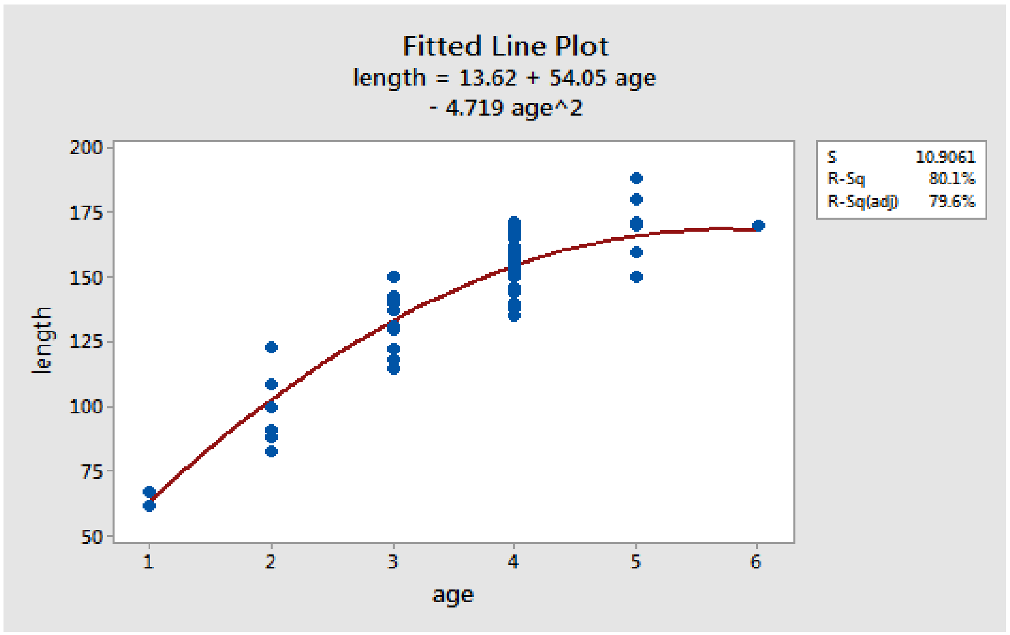


Figure 4: Polynomial Regression

$$Y=\beta 0+\beta_1 X+\beta_2 X^2+\beta_3 X^3+\epsilon$$

- Here, $\beta_0,\beta_1,\beta_2,\beta_3$ are the coefficients, and ϵ is the error term.

Characteristics:

- **Flexibility**: Polynomial regression allows for a better fit to data that has a nonlinear trend, providing more flexibility compared to simple linear regression.
- **Model Complexity**: As the degree of the polynomial increases, the curve becomes more flexible, but it can also become more complex and prone to **overfitting** (fitting the noise rather than the underlying trend).

Use Cases:

- **Nonlinear Relationships**: When the relationship between the variables is curved rather than a straight line, such as modeling the growth of plants over time or the trajectory of projectiles.
- **Trend Analysis**: Useful in analyzing trends over time when the change is not constant (e.g., sales data that increases, peaks, and then decreases).

Limitation:

- **Overfitting**: High-degree polynomials can fit the training data very well but may fail to generalize to new data.
- **Interpretability**: As the degree increases, interpreting the impact of each feature becomes more challenging.
- **Sensitivity to Outliers**: Like other regression models, polynomial regression can be sensitive to outliers, which can heavily influence the curve.

2.5 Logistic Regression:

Logistic regression is a type of regression analysis used for binary classification problems, where the outcome or dependent variable is categorical and has two possible outcomes (e.g., yes/no, true/false, or 0/1). It estimates the probability that a given input point belongs to a particular category.

Key Concepts:

- **Binary Classification**: Logistic regression is most commonly used when the response variable is binary (e.g., success/failure, spam/not spam).

- **Sigmoid Function**: Unlike linear regression, which fits a straight line to the data, logistic regression uses the sigmoid function (also called the logistic function) to model the probability of the outcome:

$$\sigma(z)=1+e-z_1$$

Where z is a linear combination of the input features:

$$z=\beta_0+\beta_1X_1+\beta_2X_2+\ldots+\beta_nX_n$$

- **Probability Output**: The output of the sigmoid function is a value between 0 and 1, representing the probability of belonging to a specific class

Mathematical Form:

The logistic regression model relates the log-odds of the probability of class membership to a linear combination of the predictors:

$$\log(1-pp)=\beta0+\beta1X1+\beta2X2+\ldots+\beta nXn$$

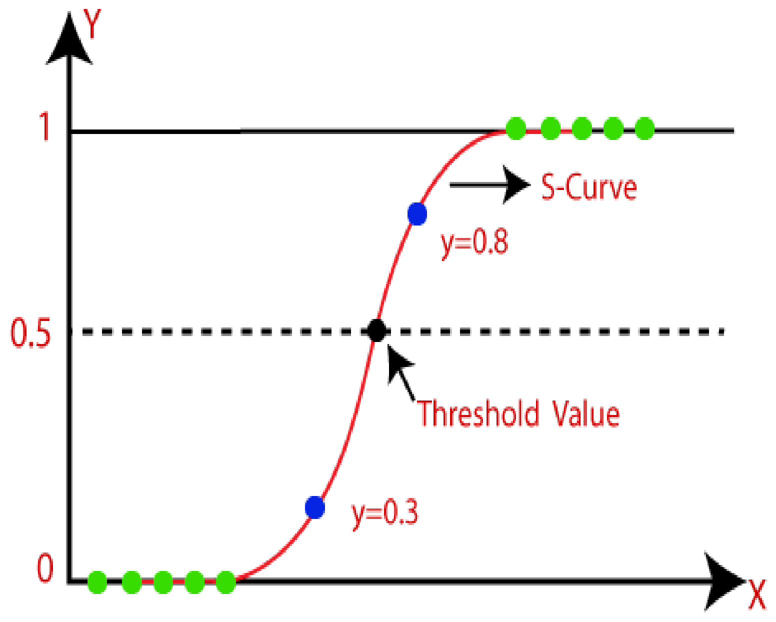


Figure 5: Logistic Regression

Where:

- p is the probability of the outcome being class 1.
- β0,β1,…,βn are the model coefficients.
- 1−pp is the **odds** of the event occurring.

Characteristics:

- **Link Function**: The logit function (log(1−pp)) is used as the link function, allowing the model to express probabilities.
- **Output**: Produces a probability estimate, which can be converted into a class prediction (e.g., p>0.5 means class 1).
- **Interpretability**: The coefficients β can be interpreted as the effect of a one-unit change in the predictor on the log-odds of the outcome.

Use Cases:

- **Medical Diagnosis**: Predicting whether a patient has a certain disease (e.g., cancer or no cancer) based on various test results.
- **Marketing**: Determining if a customer will purchase a product based on demographics and past behavior.
- **Spam Detection**: Classifying emails as spam or not spam.
- **Credit Risk Assessment**:Predicting the likelihood of a customer defaulting on a loan..

Limitations:

- **Assumption of Linearity**: Assumes a linear relationship between the log- odds of the outcome and the predictors.
- **Not Suitable for Complex Relationships**: It may not perform well when there is a complex, nonlinear relationship between the variables.
- **Sensitive to Imbalanced Data**: If one class is much more frequent than the other, it can lead to biased predictions.

2.6 Decision Tree:

A **Decision Tree** is a supervised learning algorithm used for both **classification** and **regression** tasks. It models decisions based on a tree-like structure where each internal node represents a feature (or attribute), each branch represents a decision rule, and each leaf node represents an outcome (or class). The paths from the root to the leaf represent decision rules.

- splitting a dataset into subsets based on the value of input features
- These splits are done in a way that helps to predict the target variable more accurately

Structure of a Decision Tree:

1. **Root Node**: This is the topmost node that represents the entire dataset. It splits into two or more branches.
2. **Internal Nodes**: Each internal node represents a decision point or test condition based on a feature.
3. **Branches**: They represent the outcome of the decision at each internal node, leading to further nodes or leaf nodes.
4. **Leaf Nodes**: These are the terminal nodes that provide the final outcome (classification or prediction).

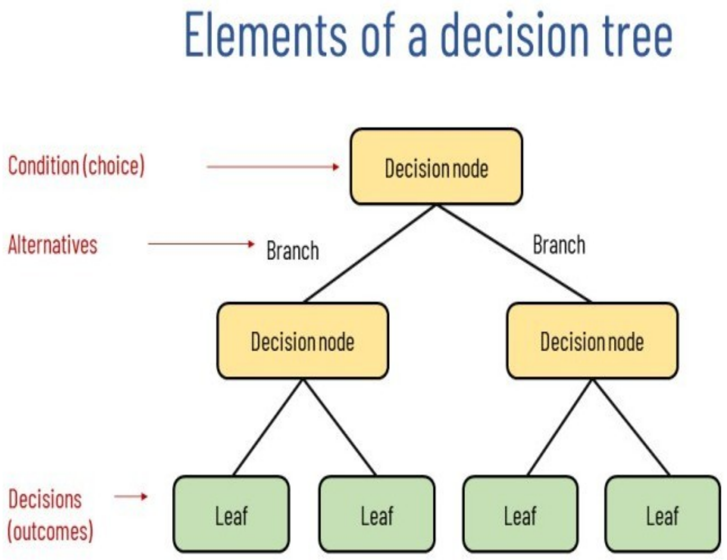


Figure 6: Decision Tree

How Decision Trees Work:

- At each node, the tree chooses a feature and a threshold value to split the data.
- The feature and threshold are selected to maximize the **purity** of the child nodes, minimizing impurity or error.
- The tree continues to split the data recursively until a stopping condition is met (e.g., maximum depth, minimum samples per leaf).

2.6.1 Types of Decision Trees:

1. Classification Trees:

- Used when the target variable is **categorical**.
- Each leaf node represents a class label, and each branch represents the decision path.
- The tree splits nodes based on measures like **Gini impurity** or **entropy** to maximize the homogeneity of the classes at each node.

Use Cases:

- Email classification as spam or not spam.
- Loan approval (approve/deny).
- Medical diagnosis (disease present/absent).

2. Regression Trees:

- Used when the target variable is **continuous**.
- Each leaf node represents a **numeric value** (the predicted value for the input).
- Splitting is based on minimizing the **Mean Squared Error (MSE)** or other measures of variance to make predictions.

Use Cases:

- Predicting house prices based on features like size, location, and number of bedrooms.
- Estimating the sales revenue for a store based on advertising spend.

Key Concepts:

- **Recursive Binary Splitting**: This is the process used to split each node into two parts, which is repeated recursively. The tree evaluates all possible splits and chooses the one that results in the lowest impurity.

- **Pruning**: To avoid **overfitting**, a decision tree can be pruned. Pruning involves removing sections of the tree that provide little additional information or that fit the noise of the training data.
- **Overfitting**: This occurs when a tree is too complex (e.g., too deep) and fits the training data too closely, performing poorly on unseen data.

Advantages of Decision Trees:

- **Simple to Understand**: The decision tree model is intuitive and easy to interpret, even without a strong mathematical background.
- **Nonlinear Relationships**: They can capture complex interactions and nonlinear relationships between features.
- **No Assumption of Feature Distribution**: They do not require data to follow a particular distribution, making them flexible for various types of data.

Disadvantages of Decision Trees:

- **Overfitting**: They are prone to overfitting, especially if not pruned properly.
- **Instability**: A small change in the data can result in a completely different tree.
- **Bias in Splitting**: Decision trees can be biased towards features with more levels (categories), which can influence the choice of splits.

2.7 Ensemble Learning:

Ensemble Learning is a machine learning technique where multiple models, often called "weak learners" or "base models," are combined to solve a particular problem. The idea is to leverage the diversity of these models to achieve better predictive performance than any single model could achieve on its own. This approach can be used for both **classification** and **regression** tasks.

Why Use Ensemble Learning?

- **Improved Accuracy**: Combining multiple models can reduce error rates and improve accuracy.
- **Robustness**: Ensembles reduce the risk of overfitting compared to using a single model, leading to more generalized predictions.
- **Reduction of Variance and Bias**: Ensemble methods can help in reducing both variance (e.g., from overfitting) and bias (e.g., from underfitting).

2.7.1 Types of Ensemble Learning Methods:

1. **Bagging (Bootstrap Aggregating)**
2. **Boosting**
3. **Stacking**

1. Bagging (Bootstrap Aggregating):

Bagging is a technique to reduce variance and prevent overfitting, particularly effective with high-variance models like decision trees.

Process:

- **Bootstrap Sampling**: Create multiple subsets of the training data by sampling with replacement.
- **Train Models**: Train a model on each subset independently.
- **Aggregation**:
  - For **regression**, take the average of all model predictions.
  - For **classification**, take the majority vote from all models.

Example:

- Random Forest is an extension of bagging where multiple decision trees are built using bootstrap samples and a random subset of features.

Example Use Case:

- Predicting house prices using multiple decision trees, each trained on different subsets of the data.

2. Boosting:

Boosting is a sequential process where each new model attempts to correct the errors made by the previous models. It focuses on **reducing bias** rather than variance.

Process:

- Train a weak learner on the training data.
- Evaluate its performance and identify errors (misclassifications or residuals).
- Emphasize the incorrect predictions in subsequent models.
- Combine the predictions of all models, often with a weighted sum approach.

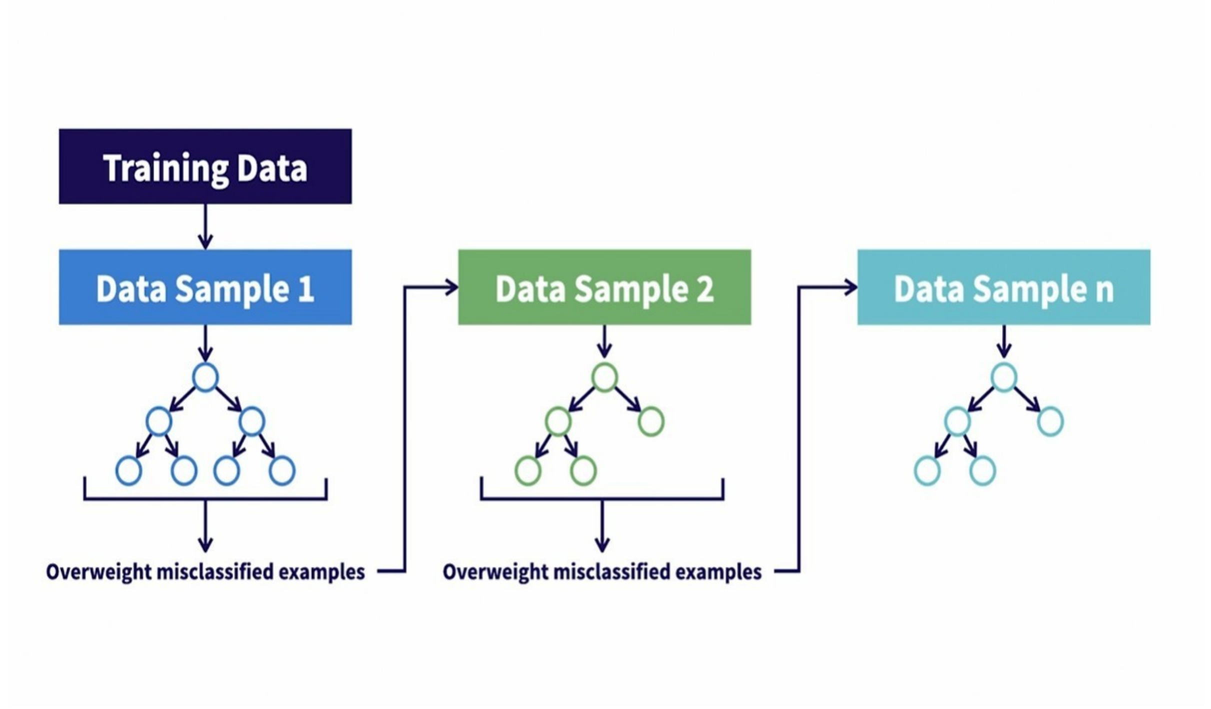

Training Data
Data Sample 1
Data Sample 2
Data Sample n
Overweight misclassified examples
Overweight misclassified examples

Figure 8: Boosting

Examples:

- **AdaBoost**: Adjusts weights of incorrectly classified instances, making them more significant for subsequent models.
- **Gradient Boosting Machines (GBM)**: Builds models iteratively, optimizing residuals using gradient descent.
- **XGBoost**: A more efficient and faster implementation of gradient boosting, especially for large datasets.

Example Use Case:

- Predicting credit risk using multiple trees that iteratively focus on misclassified loans.

3. Stacking:

Stacking, or stacked generalization, combines predictions from multiple models using a **meta-model**. It aims to leverage the strengths of each model to make better predictions.

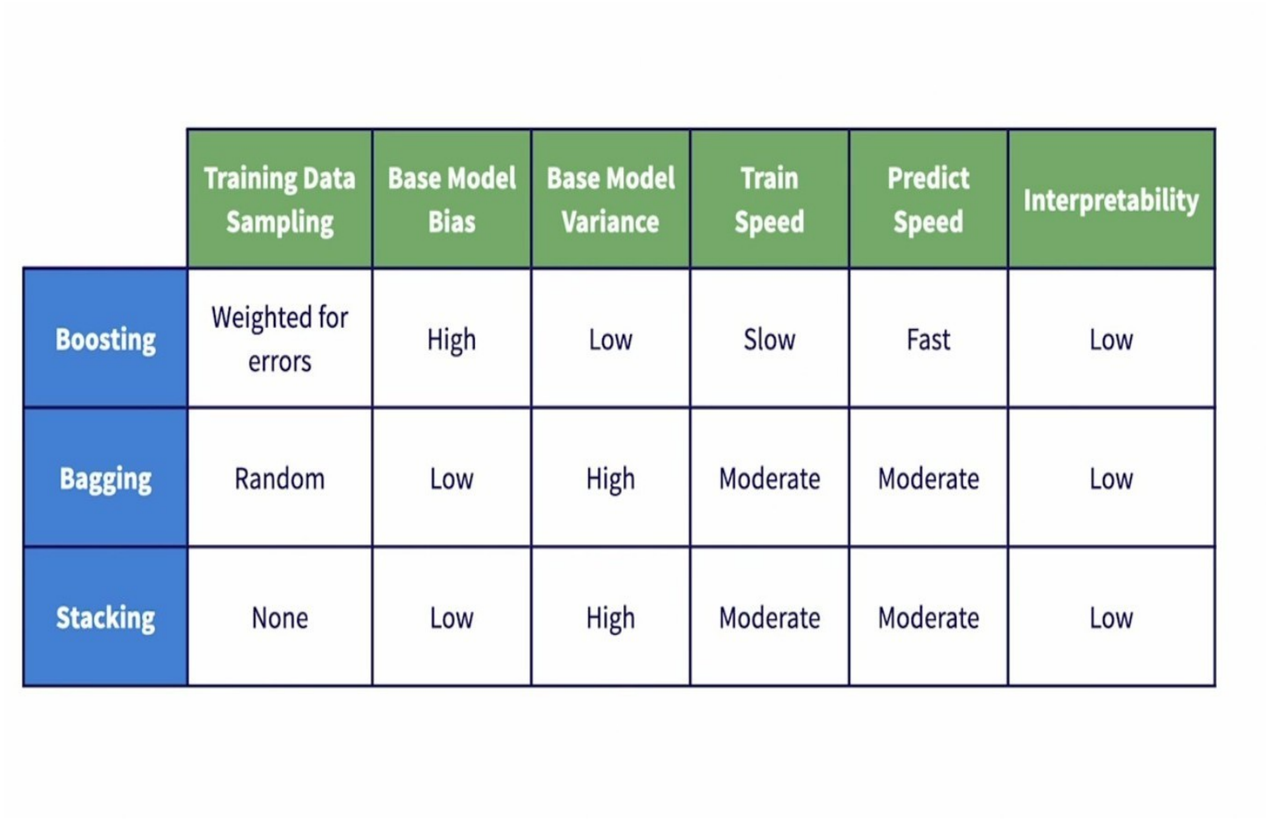
Process:

- Train multiple base models (e.g., decision trees, SVMs, logistic regression).
- Combine the predictions of these models using a new model, called a **meta-learner** (often linear regression for regression tasks or logistic regression for classification tasks).
- The meta-learner is trained on the outputs (predictions) of the base models.

Example Use Case:

- Combining different models (e.g., decision trees, support vector machines, and neural networks) to predict customer churn.

Summary of Ensemble Learning Methods



	Training Data Sampling	Base Model Bias	Base Model Variance	Train Speed	Predict Speed	Interpretability
Boosting	Weighted for errors	High	Low	Slow	Fast	Low
Bagging	Random	Low	High	Moderate	Moderate	Low
Stacking	None	Low	High	Moderate	Moderate	Low

Figure 10: Summary of Ensemble Learning Methods

Advantages of Ensemble Learning:

- **Better Performance**: Often leads to better generalization and accuracy.
- **Flexibility**: Can use a mix of different models to leverage their individual strengths.
- **Robustness**: Reduces the impact of outliers or noise in the data.

Disadvantages of Ensemble Learning:

- **Complexity**: More computationally intensive due to training multiple models.
- **Less Interpretability**: Difficult to interpret as compared to a single model.
- **Risk of Overfitting**: While ensembles generally reduce overfitting, complex ensemble models can still overfit if not tuned properly.

2.8 K-Nearest Neighbors:

- It is lazy learning algorithm used for classification and regression.
- It works by finding the 'k' nearest data points in the feature space and assigning the most common class.

Working of K-Nearest Neighbors:

- It selects k number of nearest points from given point i.e. k is 3
- It computes distance from all nearest points
- Then it selects k nearest points i.e. it has chosen 3 points as k was 3 in first step
- Then it will do majority voting
- After that it will make decision

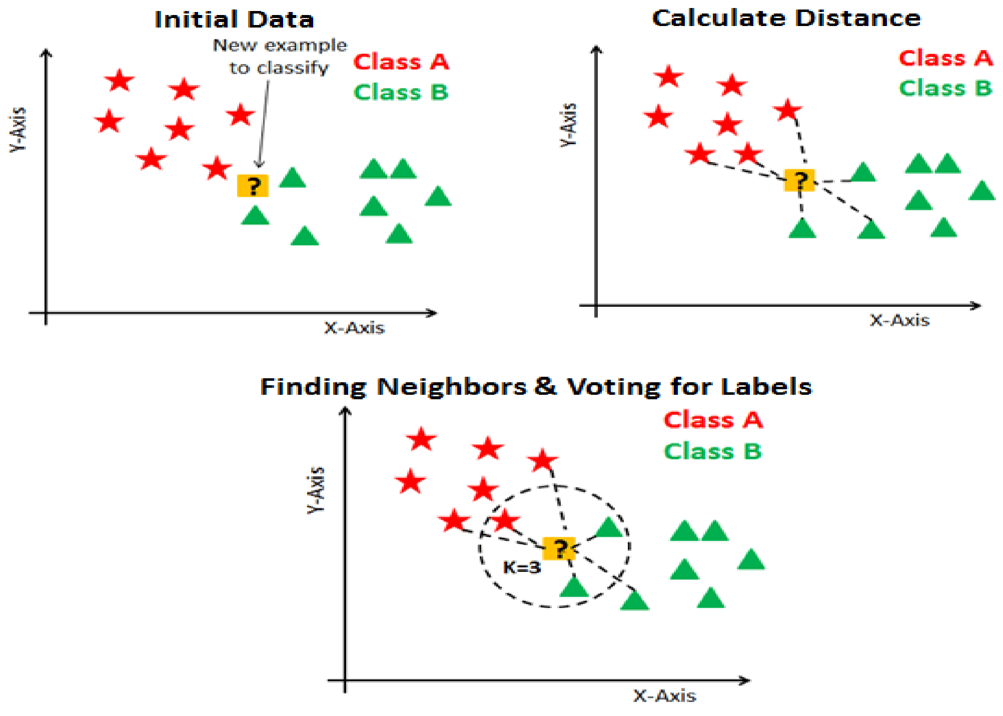


Figure 11: K-Nearest Neighbors

Pros of K-Nearest Neighbors:

- It can work in N-Dimensional Coordinates.
- KNN is one of the simplest machine learning algorithms. It doesn't require any explicit training process or model fitting
- KNN can adapt to new data easily without needing to retrain
- KNN can be used for both classification and regression tasks

Cons of K-Nearest Neighbors:

- For smaller values of k , leads to **overfitting** because of outliers.
- For higher values of k, leads to **underfitting.**
- Don't work smoothly on higher datasets.
- For higher dimensional data Euclidian distance is not measured accurately.
- In the case of an imbalanced dataset, it leads to biased decisions.

2.9 Support Vector Machines:

- It is a supervised learning algorithm used for both classification and regression tasks.
- It works by finding the optimal hyperplane that maximizes the margin between two classes
- It is the modified form of hard margin and soft margin.

Working of Support Vector Machines:

- Firstly, it understands the concept of hyperplane from the given data.
- Then it maximizes the distance between the margin as much as possible.
- Then it uses the kernel trick in case data is non-linear and difficult to understand.
- It does parameter tuning
- Finally, it does model training and prediction.

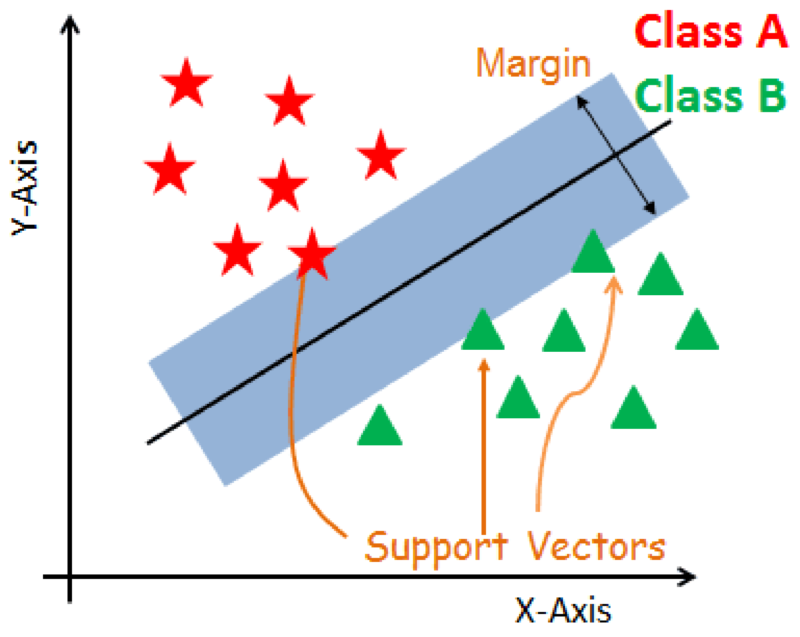


Figure 12: Support Vector Machines

Types of data:

There are two types of data.

- Linear data: It can easily be separated into classes
- Non-Linear: It is complex data and cannot be easily separated into classes. SVM uses kernel trick for this type of data.

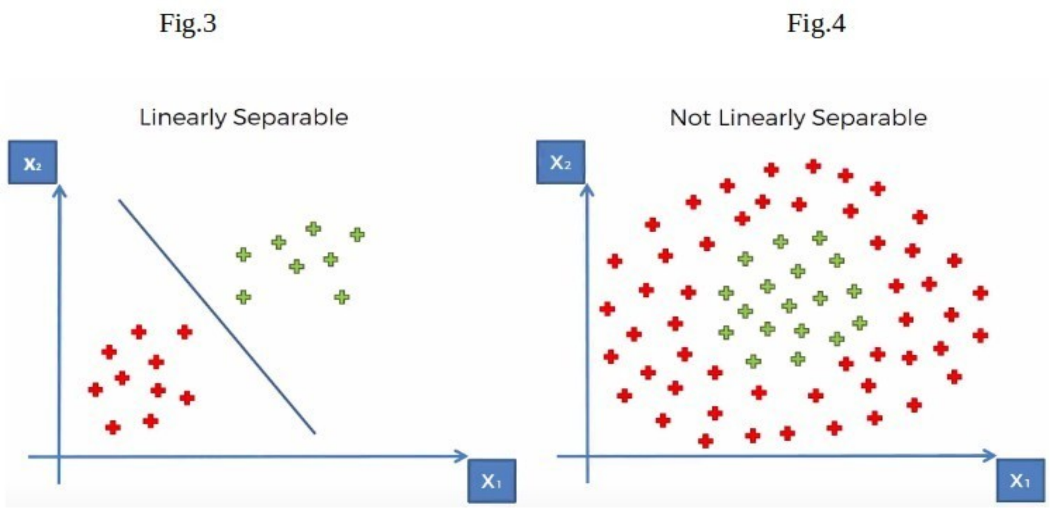


Figure 13: Types of Support Vector Machines

Pros of Support Vector Machines:

- It works best on linear data, non-linear data too because of hard margin, soft margin and kernel trick.
- SVM is effective when the number of features is greater than the number of samples. It can handle high-dimensional data very well.
- It reduces the chances of **overfitting**.
- The mathematical basis of SVM is very strong.

Cons of Support Vector Machines:

- It is very difficult to understand because the mathematics behind SVM is very complex.
- In hard margin, there is problem of dealing with non-linear data.
- In soft margin, there is problem of how long margin should be.
- There is difficulty in choosing the appropriate kernel function and it requires domain knowledge.

2.10 Linear Discriminant Analysis:

- It is a supervised machine learning technique used in dimensionality reduction.
- It projects feature in high dimensional space into low dimension

Working of Linear Discriminant Analysis:

- Firstly, it calculates the **mean** of all classes present in labelled data.
- Then, it derives a **covariance matrix** of class variables.
- Then, it calculates the within and between scatter matrixes.
- It computes eigen values from within and between scatter matrixes.
- It sorts the eigen values in descending order.
- Finally, it computes the eigen vector and obtain the linear discriminants by taking the dot products of eigen vectors and original data.

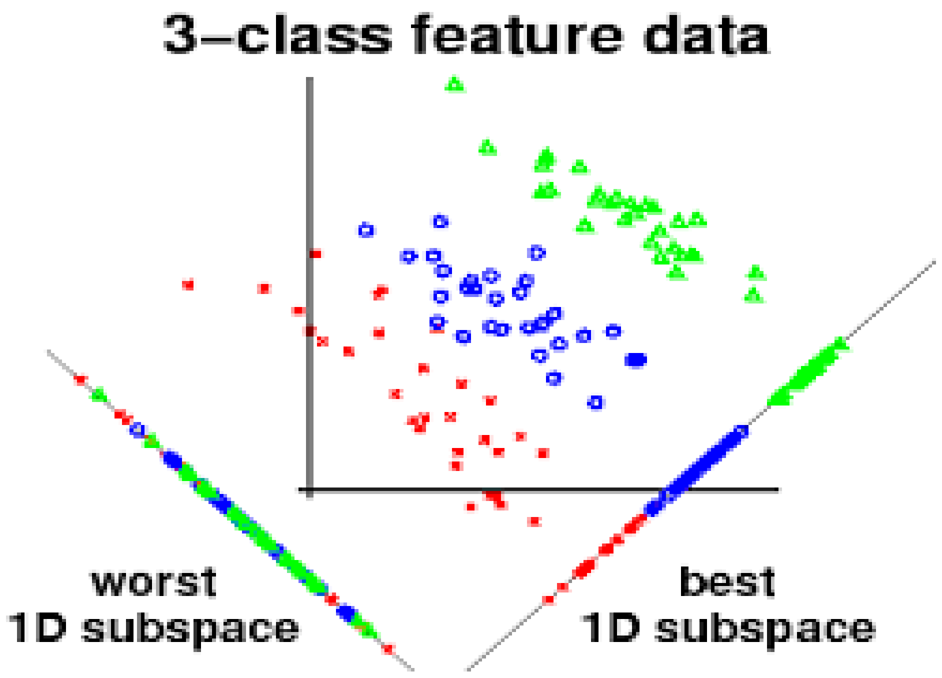


Figure 14: Linear Discriminant Analysis

Pros of Linear Discriminant Analysis:

- It effectively reduces features redundancy.
- It works well in **dimensionality reduction**.
- It reduces overfitting.
- It involves simple linear algebra, making it computationally efficient and fast.
- It provides clear decision boundaries and is easy to interpret.

Cons of Linear Discriminant Analysis:

- It assumes that data has **gaussian distribution**, which is not correct for every case.
- It also assumes classes are linearly separated which may not be true.
- It is sensitive to **outliers** because it relies on mean and variance value of data.
- It is a slow algorithm.
- Similarly, it struggles with imbalanced data sets.

3 ROOT:

ROOT is a powerful software framework developed by CERN for data analysis, particularly tailored for high-energy physics experiments. It offers a comprehensive set of tools for managing, analyzing, and visualizing large datasets, making it standard in the field of particle physics.

What is ROOT?

- **ROOT**: Stands for "ROOT Object-Oriented Technology" and is written primarily in C++ but supports Python through PyROOT.
- **Developed by**: CERN, the European Organization for Nuclear Research.
- **Purpose**: To analyze particle physics data, especially large-scale data produced by detectors such as those in the Large Hadron Collider (LHC).

Uses of ROOT:

ROOT is designed for various tasks in particle physics data analysis, including:

- **Data Storage and Management**: Handles large datasets using efficient I/O operations. It can store data in trees, histograms, and graphs.

- **Data Visualization**: Provides tools for creating 1D, 2D, and 3D histograms, scatter plots, and other visualizations.

- **Statistical Analysis**: Performs statistical tests, curve fitting, and data modeling.

- **Machine Learning**: Integrates machine learning methods through the TMVA (Toolkit for Multivariate Analysis) for tasks like classification and regression.

3.1 Key Libraries in ROOT:

ROOT's functionality is organized into several libraries, with some of the most important being:

3.1.1 TH1F, TH2F, TH3F (Histograms):

These classes are used for creating and managing histograms in ROOT.

- **TH1F**: 1D histogram (e.g., distribution of particle energy).

- **TH2F**: 2D histogram (e.g., correlation between two variables like momentum and angle).
- **TH3F**: 3D histogram (e.g., 3D spatial distribution of particles).

3.1.2 Ttree:

- Represents a structured way to store and access large amounts of data.
- Each "branch" in a tree holds a variable, and each "entry" is an event or observation.
- Useful for large datasets like particle physics experiments where data is stored as events.

3.1.3 TMVA (Toolkit for Multivariate Analysis):

- A library for applying machine learning algorithms in ROOT.
- Supports methods like decision trees, neural networks, and more.
- Helps in classification problems (e.g., identifying particles) and regression tasks.

4 Exploratory Data Analysis:

In this phase, I have been provided with data in ROOT format, stored in a TTree structure. The goal is to analyze this data, but before diving into the analysis, it is crucial to understand the nature of the data and its structure. The data consists of a SignalTree and several BackgroundTrees representing different physics processes.

4.1 Data Structure Overview:

SignalTree:

Represents events where four Top-W bosons are produced, which is the signal of interest in this analysis.

Background Trees:

- **ttbarBackground**: Represents events with top-antitop (ttbar) production, which can mimic the signal.

- **ttbarHBackground**: Events involving top-antitop production with an associated Higgs boson.
- **ttbarWBackground**: Events involving top-antitop production with an associated W boson.
- **ttbarZBackground**: Events involving top-antitop production with an associated Z boson.
- **tripletopwmBackground**: Events with three top quarks and a W- boson.
- **tripletopwpBackground**: Events with three top quarks and a W+ boson.
- **tripletopbbarBackground**: Events with three top quarks and a bottom-antibottom pair.

Weighted Trees:

Each background and signal tree has a weighted version, which adjusts event counts to reflect their likelihood of occurrence.

The data in each of these trees includes 21 variables that describe various physical properties of the events.

Variables in Each Tree:

1. **jet1_pt, jet2_pt, ..., jet6_pt**: Transverse momentum (pT) of up to six jets in the event, indicating the momentum of the jets perpendicular to the beam axis.

2. **numJets**: Total number of jets observed in the event, providing information on the event's complexity.

3. **numBJets**: Number of b-tagged jets (jets likely originating from b-quarks), important for identifying decays involving top quarks.

4. **bjet1_pt**: Transverse momentum of the leading b-jet (the most energetic b- tagged jet).

5. **sumbjet_pt**: Sum of the transverse momenta of all b-jets, indicating the overall energy associated with b-jets in the event.

6. **Scalar_HT**: The scalar sum of the transverse momenta of all jets, representing the total transverse energy in the event.

7. **missing_ET**: Magnitude of missing transverse energy (MET), reflecting energy imbalance in the transverse plane, often due to undetected neutrinos.

8. **rat_MET_HT**: Ratio of missing transverse energy to scalar HT, used to distinguish between signal and background processes.

9. **deltaEta_jets**: Difference in pseudorapidity (η) between the leading jets, useful for characterizing jet spread in the event.

10. **deltaPhi_jets**: Difference in azimuthal angle (ϕ) between the leading jets, indicating angular separation in the transverse plane.

11. **deltaEta_bjets**: Difference in pseudorapidity between b-jets, relevant for understanding the distribution of b-jets.

12. **deltaPhi_bjets**: Difference in azimuthal angle between b-jets, helping to analyze their relative positions.

13. **deltaR_jets**: The distance measure between jets in (η,ϕ) space, combining deltaEta and deltaPhi to describe spatial separation.

14. **deltaR_bjets**: Similar to deltaR_jets but specifically for b-jets, providing insight into b-jet clustering.

15. **top_mass**: Reconstructed mass of the top quark, essential for identifying top quark events.

16. **Weight**: Event weight, which scales events according to their occurrence probability to better reflect the underlying physical processes.

With a clear understanding of the data structure and variables, this initial step of data analysis lays the foundation for exploring and comparing the signal and background events.

4.2 Analysis Report: Differentiating Signal and Background Events Using Machine Learning Techniques

Introduction:

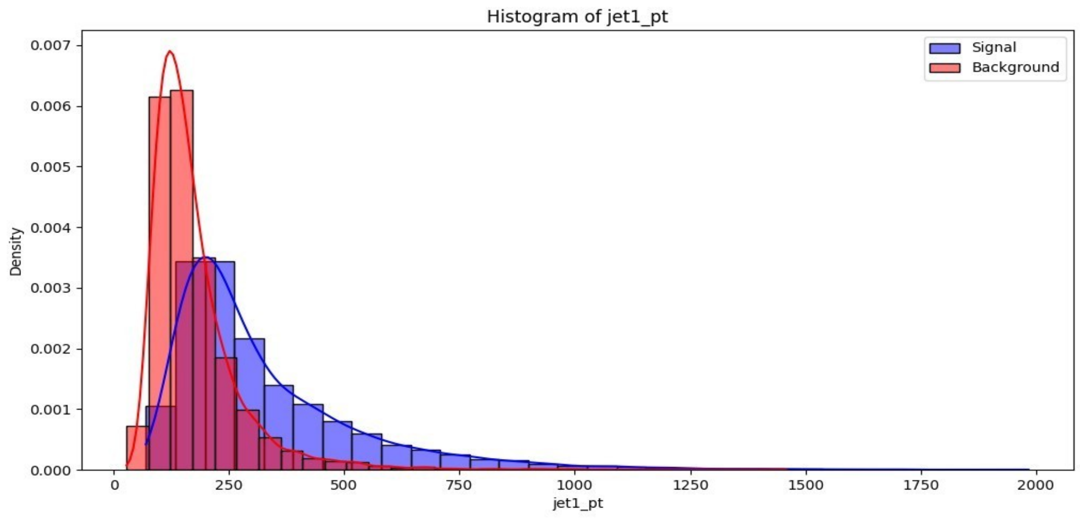
In this analysis, we aim to differentiate between signal and background events in particle physics data using a combination of statistical plots and machine learning methods. This report will provide a concise overview of the steps taken, the insights gained from each type of analysis, and the key variables that contribute to the classification.

Step 1: Histogram Analysis:

Purpose:

Histograms are used to visualize the distribution of individual variables for both signal and background events. They help us identify key differences in the distributions of variables between the two classes, which is crucial for selecting features that could improve classification.

Graph:



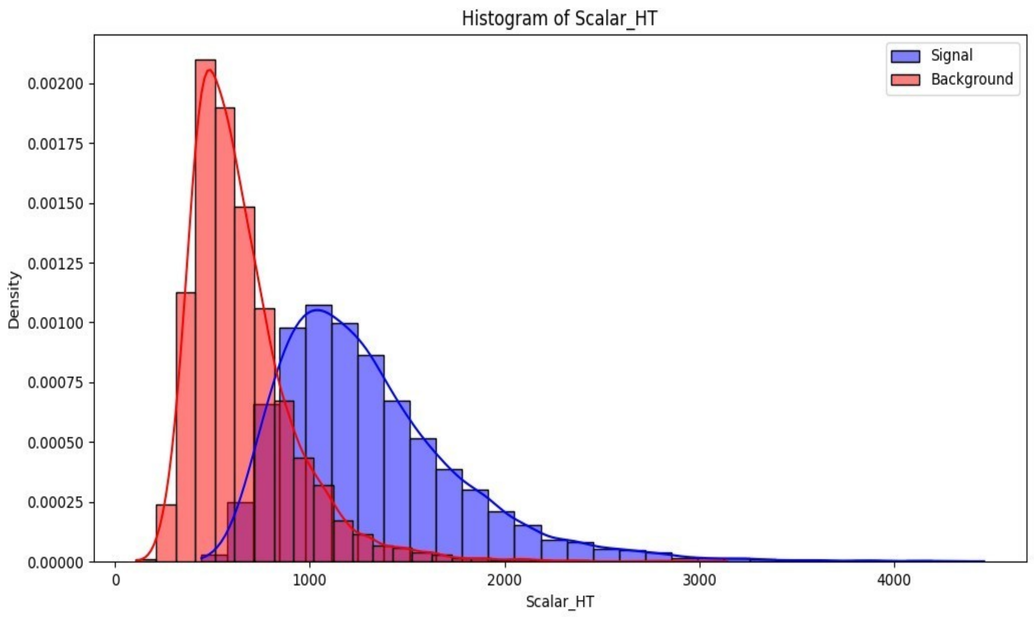


Figure 17: Histogram of Scalar_HT

Insights from Histograms:

- **jet1_pt**: Signal events tend to have higher values for the momentum of the first jet compared to background events.
- **numJets**: Signal events generally involve a higher number of jets.
- **Scalar_HT**: The total transverse momentum (HT) is significantly higher for signal events than for background events.
- These insights help in identifying variables where the separation between signal and background is most prominent.

Step 2: Box Plot Analysis:

Purpose:

Box plots provide a summary of the distribution of variables, including the median, interquartile range (IQR), and potential outliers. They are useful for understanding the spread and variability of each variable, highlighting any skewness or unusual data points.

Graph:

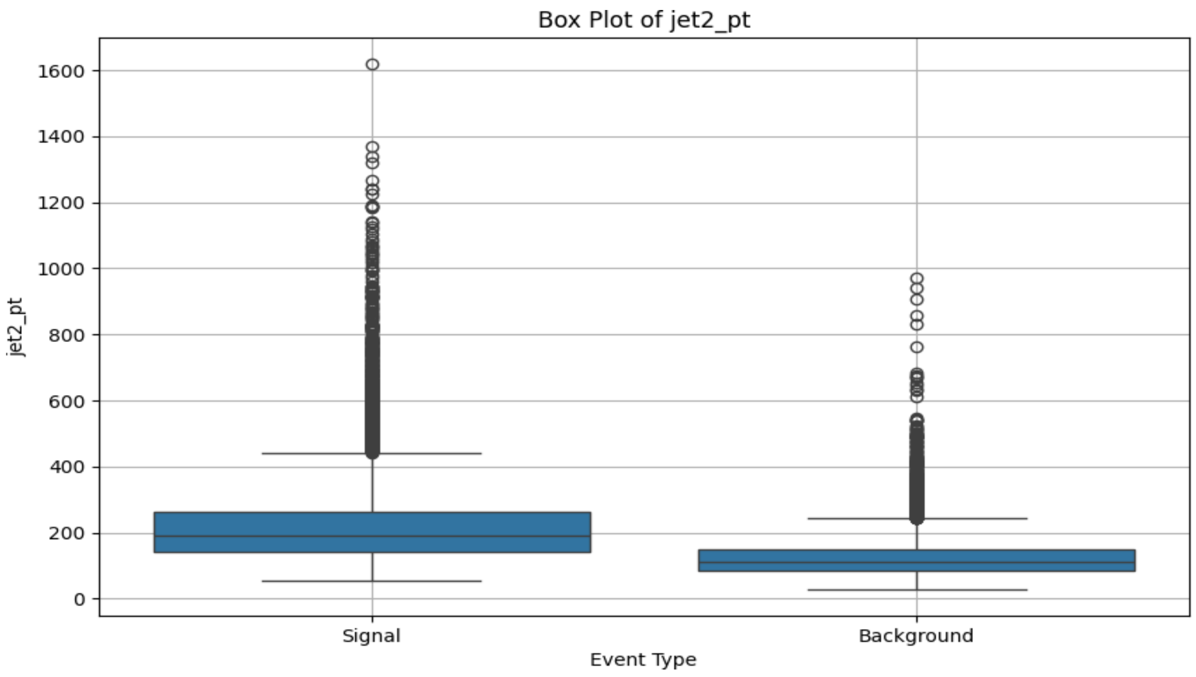


Figure 18: Box Plot of jet2_pt

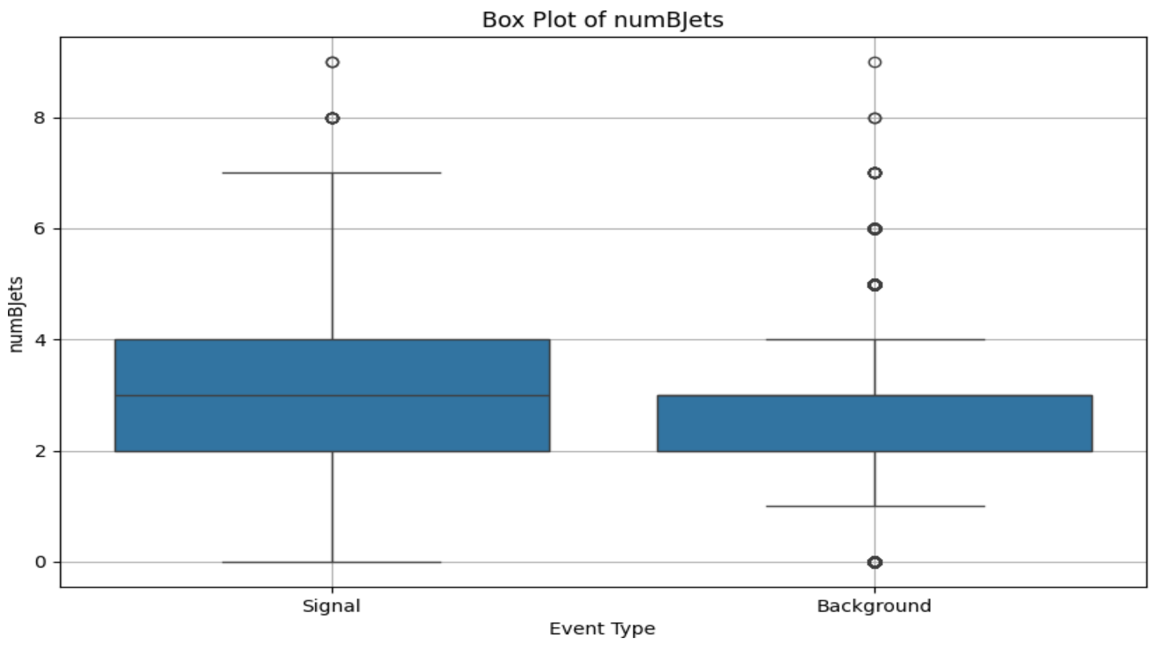


Figure 19: Box Plot of numBjets

Insights from Box Plots:

- **jet2_pt**: The interquartile range (IQR) for signal is wider, indicating greater variability compared to the background.
- **numBJets**: Signal events show a slightly higher number of b-tagged jets than background events, but with significant overlap.
- Box plots reinforce the histogram findings, showing the differences in the spread of values for each variable.

Step 3: Density Plot Analysis:

Purpose:

Density plots are used to estimate the probability density of a continuous variable. They help in visualizing the smooth distribution of variables and identifying where the signal and background overlap or diverge.

Graph:

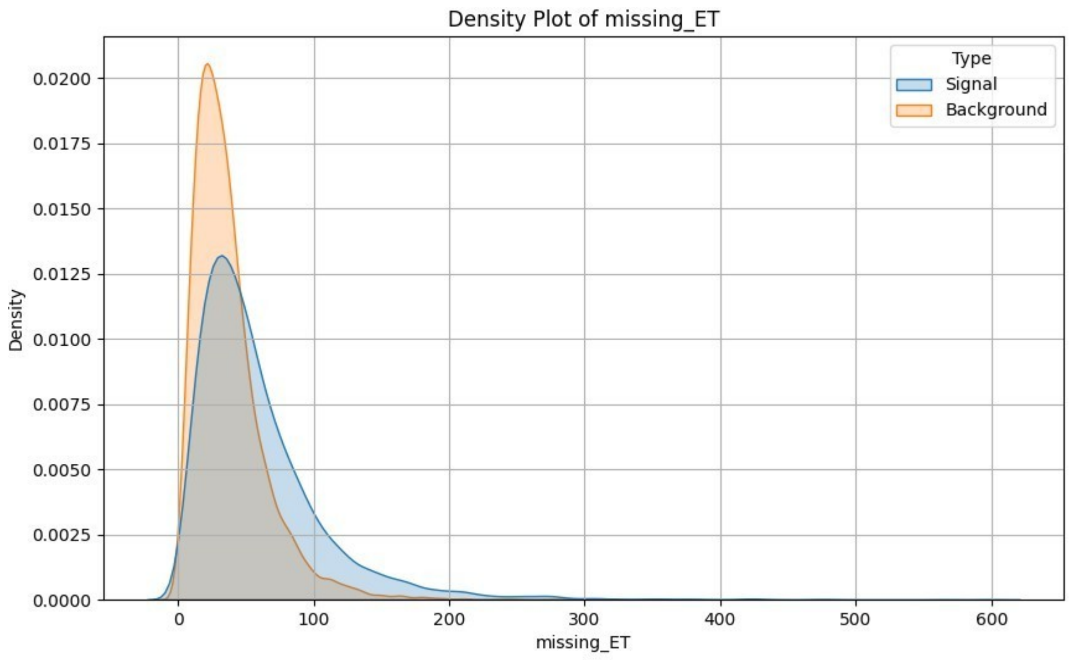


Figure 20: Density Plot of missing_ET

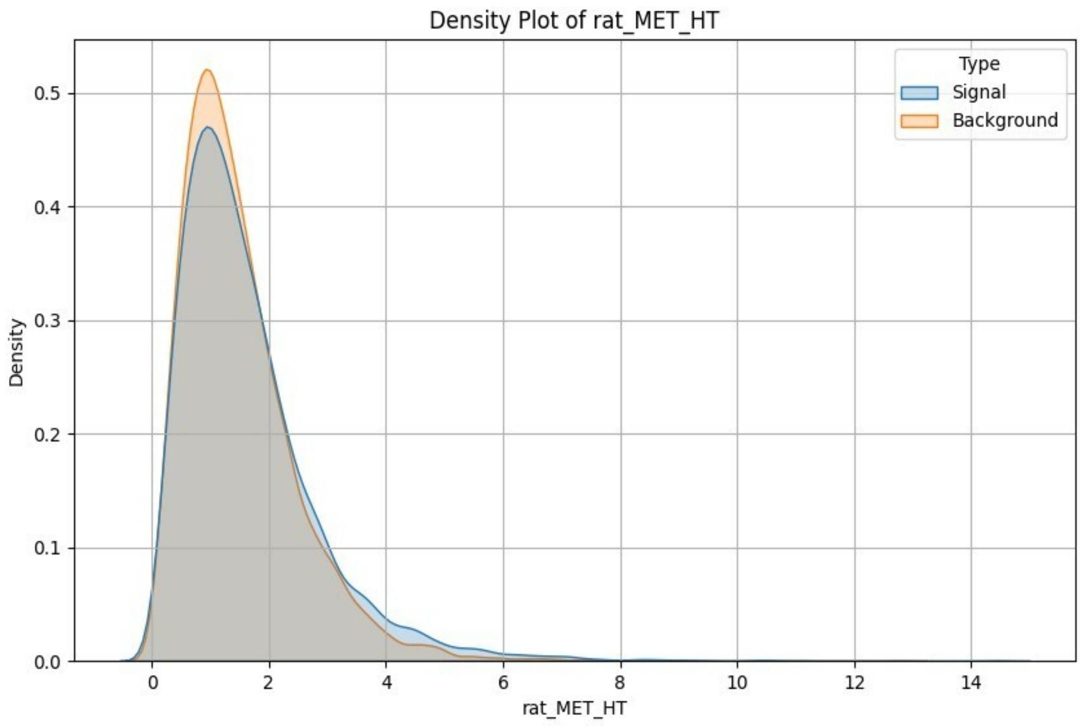


Figure 21: Density Plot of rat_MET_HT

Insights from Density Plots:

- **missing_ET**: Density plots show that signal events tend to have higher missing transverse energy than background events.
- **rat_MET_HT**: The ratio of missing energy to total transverse momentum helps in further distinguishing between signal and background events, but with some overlap.
- The density plots provide a clear visualization of how each variable's distribution overlaps between signal and background.

Step 4: Feature Importance Analysis Using Random Forest:

Purpose:

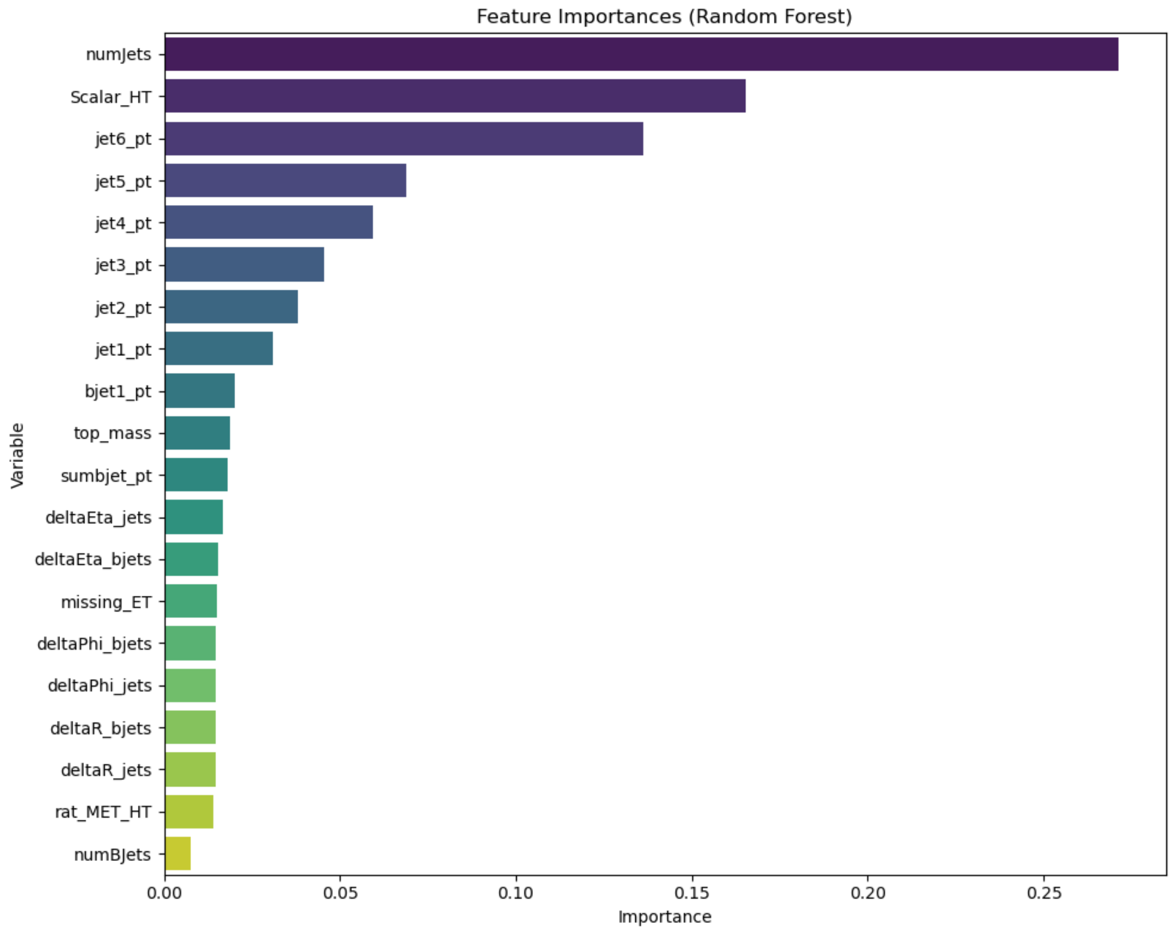
A Random Forest is an ensemble learning method that builds multiple decision trees and combines their results to improve the accuracy and stability of predictions. Random Forest Classifier helps in determining the importance of each variable in distinguishing between signal and background events. This insight allows us to focus on the most impactful variables for further analysis.

Top Variables (by Importance):

- **numJets** - The total number of jets in an event is the most influential feature. High counts of jets can indicate more complex or high-energy interactions, typically associated with signal events.

- **Scalar_HT** - This represents the scalar sum of transverse momenta (pT) of all jets in the event. Higher values of Scalar_HT often signal high-energy processes, making it valuable for identifying signals.
- **jet6_pt** - The transverse momentum (pT) of the sixth jet in the event is a key factor. The pT of higher-order jets can be crucial in differentiating events with multiple jets, especially for complex processes.
- **jet5_pt** - The transverse momentum of the fifth jet is also influential. Its inclusion highlights that the characteristics of multiple jets (beyond just the primary few) play a role in identifying signal events.
- **jet4_pt** - The pT of the fourth jet further adds detail about the energy distribution among jets. It provides additional context on jet activity, useful for distinguishing signal and background interactions.

These variables together contribute to the model's ability to classify events accurately by capturing essential aspects of jet count, energy, and momentum, critical to the specific physics analysis being conducted.



Step 5: Pairwise Scatter Plots and Heatmap Analysis:

Purpose:

Pairwise scatter plots and heatmaps of correlations help us understand relationships between different variables. This step is essential for detecting any multicollinearity between features and for visualizing how combinations of variables differ between signal and background.

Graph:

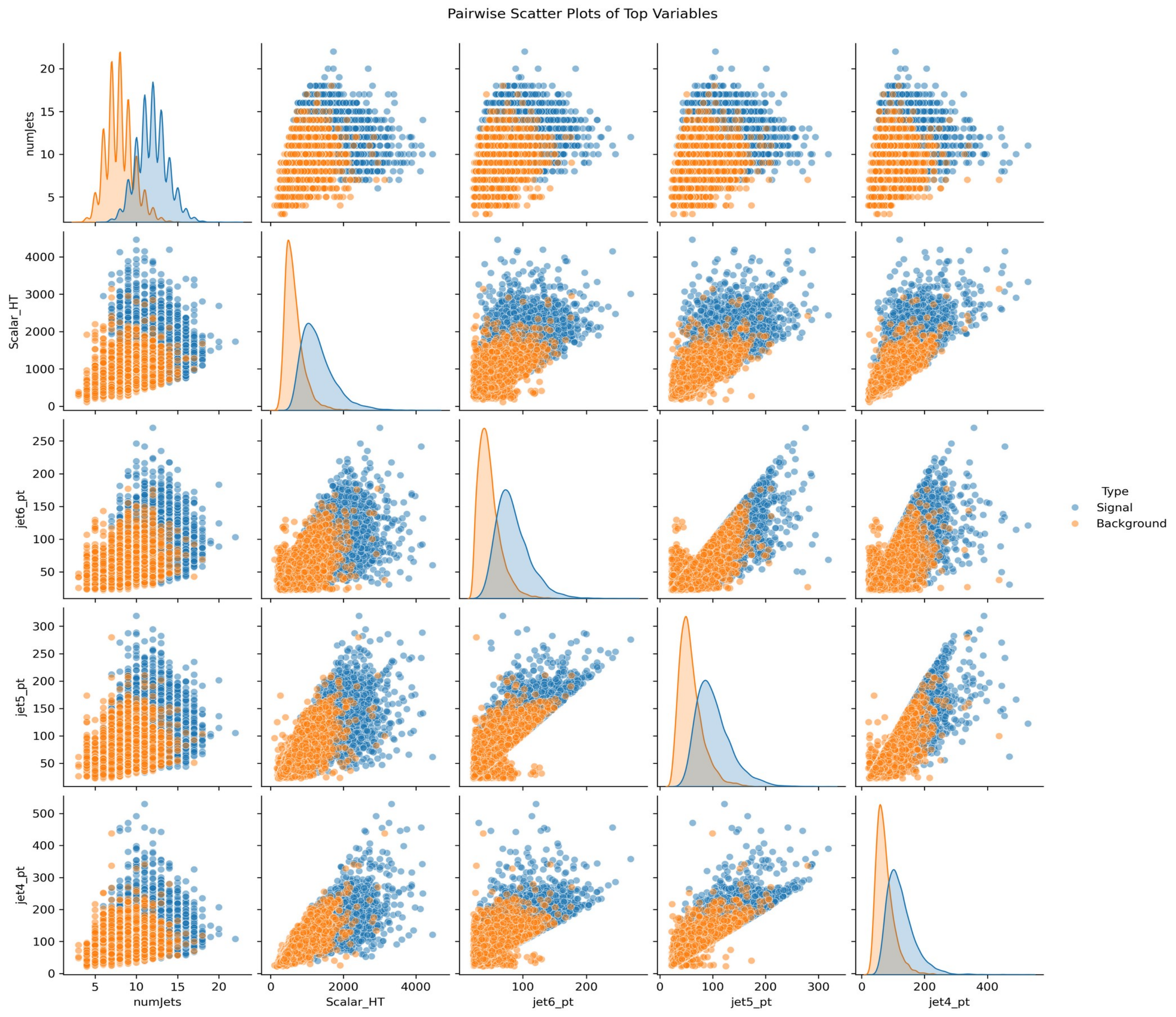
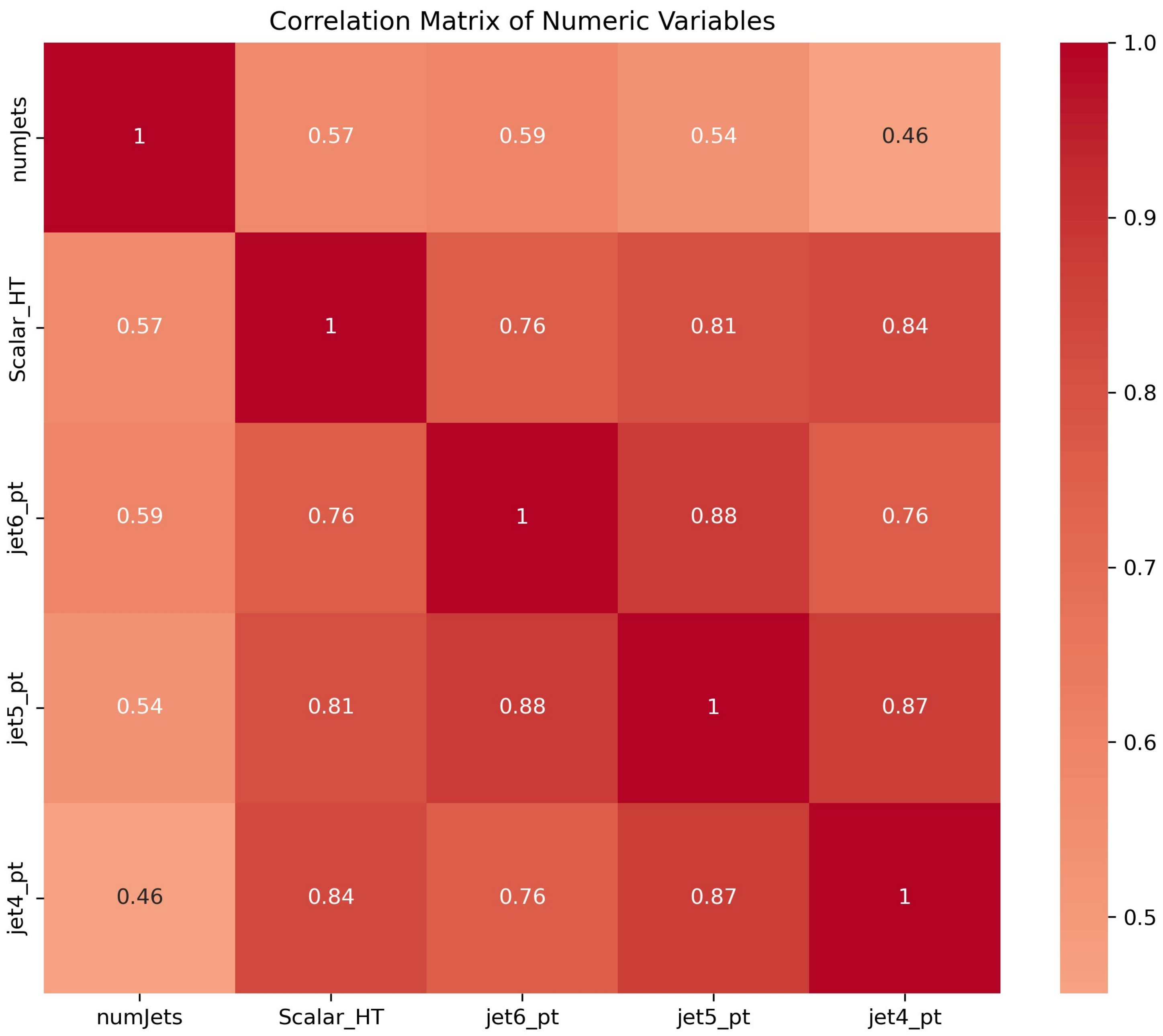


Figure 22: Pairwise Scatter Plots



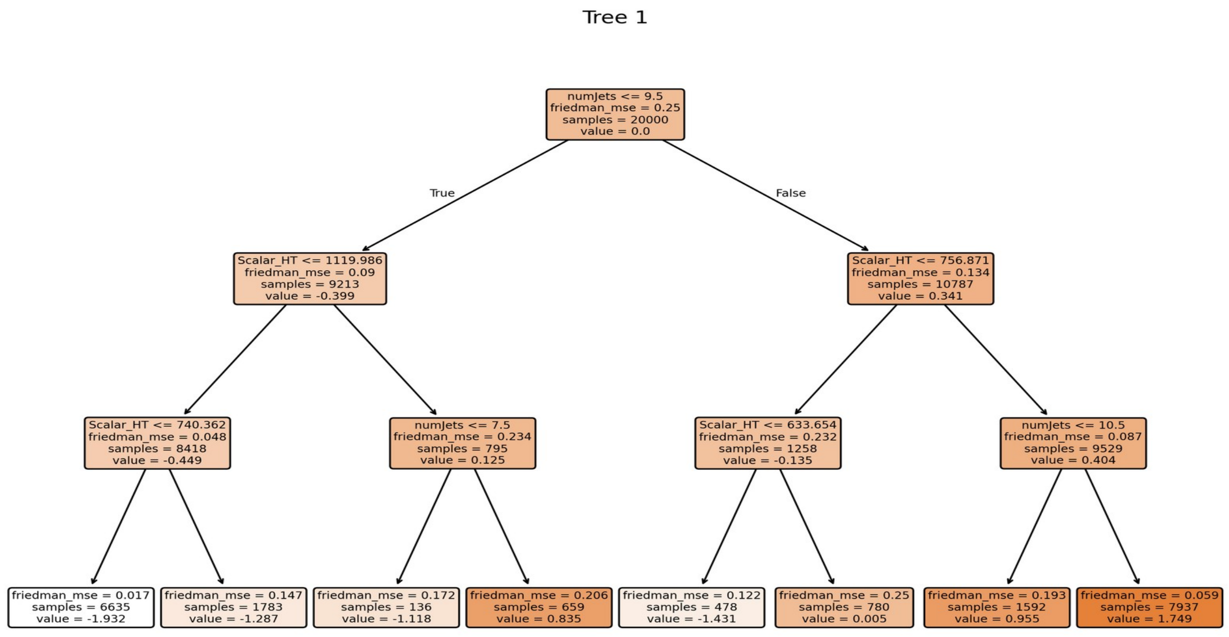
Step 5: Boosted Decision Tree Analysis on Full Dataset:

Purpose:
A Boosted Decision Tree, specifically the Gradient Boosting Classifier, is an ensemble learning method that builds a series of weak decision trees, each

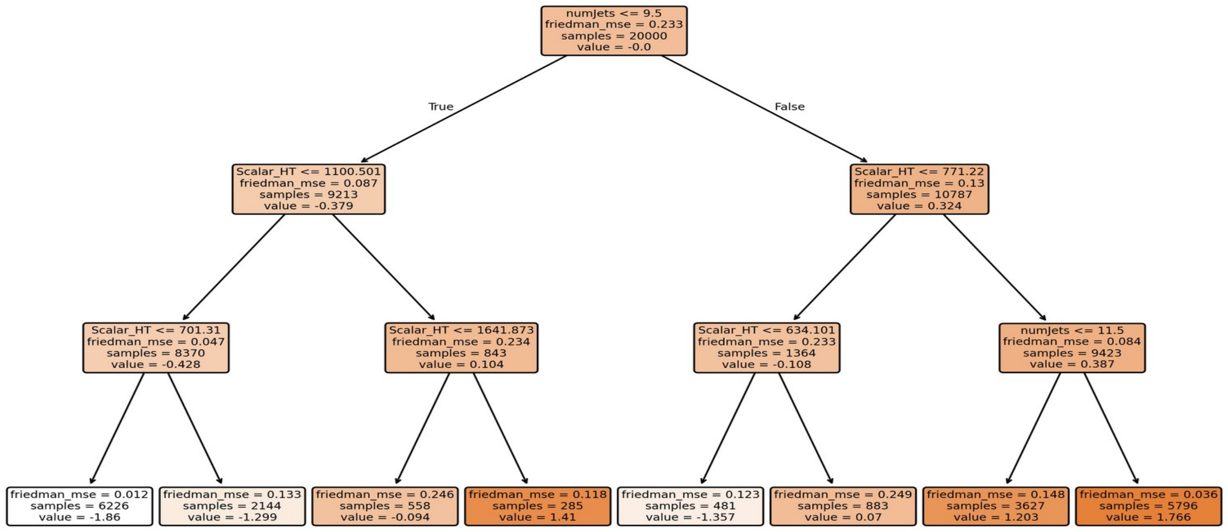
attempting to correct the errors of the previous one. This process allows the model to focus on the most challenging cases. The purpose of applying this technique is to classify events into signal (1) and background (0) categories with high accuracy. The model achieves better generalization and predictive performance by leveraging the combined strength of multiple trees trained on different subsets of the data.

Model Configuration:

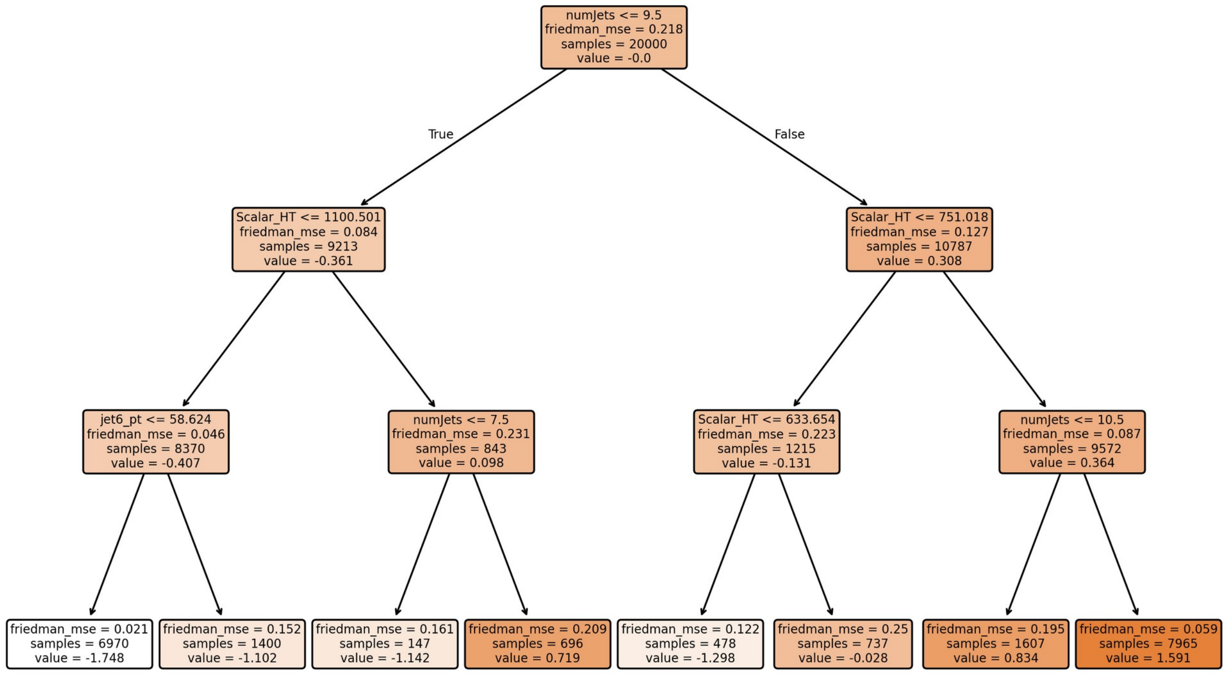
- **n_estimators=100:** A higher number of trees is used to ensure that the model has sufficient capacity to correct errors and capture the complex relationships within the data.
- **learning_rate=0.05:** The learning rate is kept small to ensure the model learns slowly and avoids overfitting by not allowing large adjustments in each iteration.
- **max_depth=3:** The decision trees are kept shallow to prevent overfitting, focusing on general patterns rather than complex, potentially noise-driven splits.
- **min_samples_split=10** and **min_samples_leaf=5:** These parameters ensure that nodes need a minimum number of samples to be split and leaves have enough samples to make stable decisions, further aiding in generalization.
- **subsample=0.8:** Random sampling of 80% of the data for each tree reduces the chance of overfitting and increases model robustness by introducing some randomness.



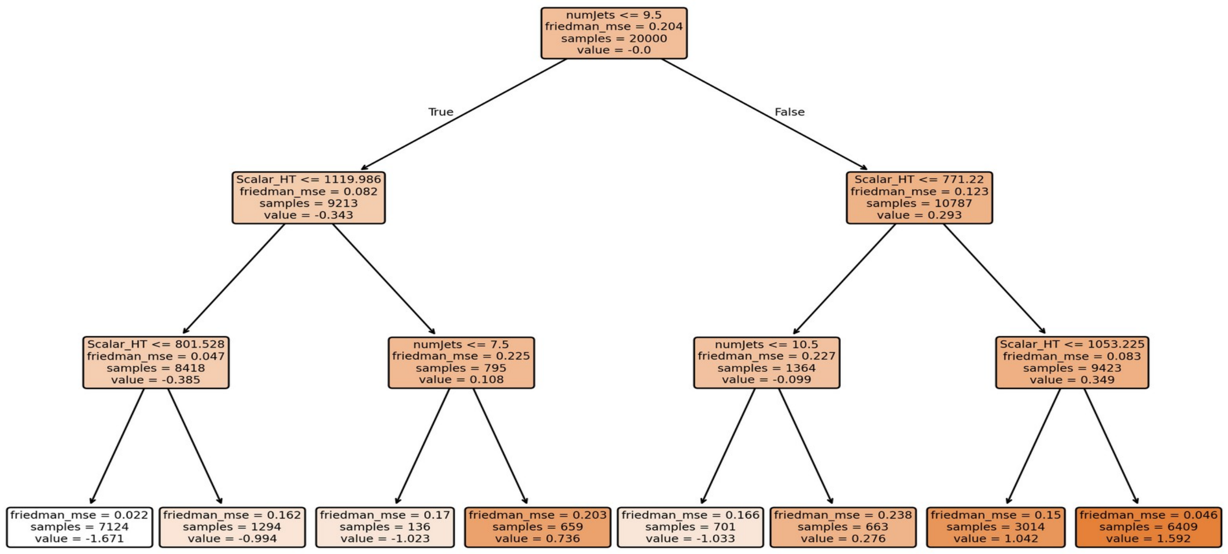
Tree 2


numJets <= 9.5
friedman_mse = 0.233
samples = 20000
value = -0.0
True
False
Scalar_HT <= 1100.501
friedman_mse = 0.087
samples = 9213
value = -0.379
Scalar_HT <= 771.22
friedman_mse = 0.13
samples = 10787
value = 0.324
Scalar_HT <= 701.31
friedman_mse = 0.047
samples = 8370
value = -0.428
Scalar_HT <= 1641.873
friedman_mse = 0.234
samples = 843
value = 0.104
Scalar_HT <= 634.101
friedman_mse = 0.233
samples = 1364
value = -0.108
numJets <= 11.5
friedman_mse = 0.084
samples = 9423
value = 0.387
friedman_mse = 0.012
samples = 6226
value = -1.86
friedman_mse = 0.133
samples = 2144
value = -1.299
friedman_mse = 0.246
samples = 558
value = -0.094
friedman_mse = 0.118
samples = 285
value = 1.41
friedman_mse = 0.123
samples = 481
value = -1.357
friedman_mse = 0.249
samples = 883
value = 0.07
friedman_mse = 0.148
samples = 3627
value = 1.203
friedman_mse = 0.036
samples = 5796
value = 1.766

Tree 3

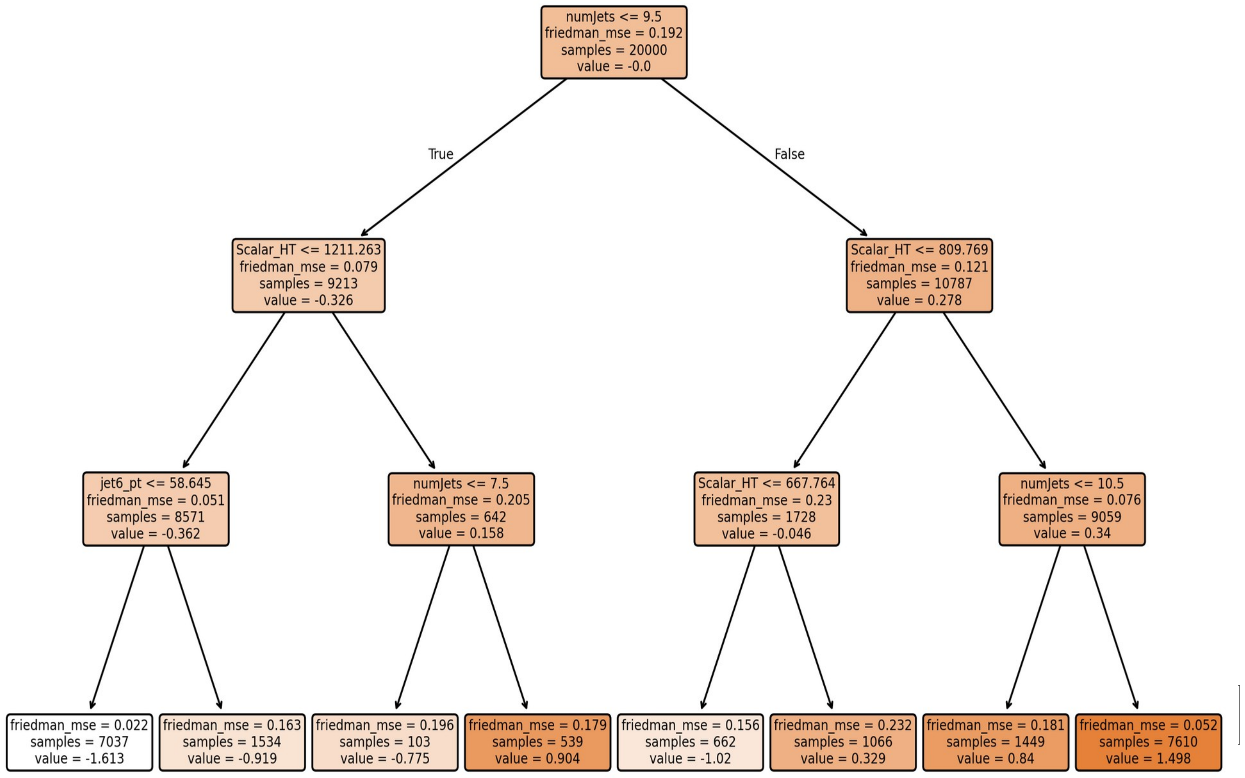

numJets <= 9.5
friedman_mse = 0.218
samples = 20000
value = -0.0
True
False
Scalar_HT <= 1100.501
friedman_mse = 0.084
samples = 9213
value = -0.361
Scalar_HT <= 751.018
friedman_mse = 0.127
samples = 10787
value = 0.308
jet6_pt <= 58.624
friedman_mse = 0.046
samples = 8370
value = -0.407
numJets <= 7.5
friedman_mse = 0.231
samples = 843
value = 0.098
Scalar_HT <= 633.654
friedman_mse = 0.223
samples = 1215
value = -0.131
numJets <= 10.5
friedman_mse = 0.087
samples = 9572
value = 0.364
friedman_mse = 0.021
samples = 6970
value = -1.748
friedman_mse = 0.152
samples = 1400
value = -1.102
friedman_mse = 0.161
samples = 147
value = -1.142
friedman_mse = 0.209
samples = 696
value = 0.719
friedman_mse = 0.122
samples = 478
value = -1.298
friedman_mse = 0.25
samples = 737
value = -0.028
friedman_mse = 0.195
samples = 1607
value = 0.834
friedman_mse = 0.059
samples = 7965
value = 1.591

Tree 4



numJets <= 9.5
friedman_mse = 0.204
samples = 20000
value = -0.0
True
False
Scalar_HT <= 1119.986
friedman_mse = 0.082
samples = 9213
value = -0.343
Scalar_HT <= 771.22
friedman_mse = 0.123
samples = 10787
value = 0.293
Scalar_HT <= 801.528
friedman_mse = 0.047
samples = 8418
value = -0.385
numJets <= 7.5
friedman_mse = 0.225
samples = 795
value = 0.108
numJets <= 10.5
friedman_mse = 0.227
samples = 1364
value = -0.099
Scalar_HT <= 1053.225
friedman_mse = 0.083
samples = 9423
value = 0.349
friedman_mse = 0.022
samples = 7124
value = -1.671
friedman_mse = 0.162
samples = 1294
value = -0.994
friedman_mse = 0.17
samples = 136
value = -1.023
friedman_mse = 0.203
samples = 659
value = 0.736
friedman_mse = 0.166
samples = 701
value = -1.033
friedman_mse = 0.238
samples = 663
value = 0.276
friedman_mse = 0.15
samples = 3014
value = 1.042
friedman_mse = 0.046
samples = 6409
value = 1.592

Tree 5



numJets <= 9.5
friedman_mse = 0.192
samples = 20000
value = -0.0
True
False
Scalar_HT <= 1211.263
friedman_mse = 0.079
samples = 9213
value = -0.326
Scalar_HT <= 809.769
friedman_mse = 0.121
samples = 10787
value = 0.278
jet6_pt <= 58.645
friedman_mse = 0.051
samples = 8571
value = -0.362
numJets <= 7.5
friedman_mse = 0.205
samples = 642
value = 0.158
Scalar_HT <= 667.764
friedman_mse = 0.23
samples = 1728
value = -0.046
numJets <= 10.5
friedman_mse = 0.076
samples = 9059
value = 0.34
friedman_mse = 0.022
samples = 7037
value = -1.613
friedman_mse = 0.163
samples = 1534
value = -0.919
friedman_mse = 0.196
samples = 103
value = -0.775
friedman_mse = 0.179
samples = 539
value = 0.904
friedman_mse = 0.156
samples = 662
value = -1.02
friedman_mse = 0.232
samples = 1066
value = 0.329
friedman_mse = 0.181
samples = 1449
value = 0.84
friedman_mse = 0.052
samples = 7610
value = 1.498

Performance:

Metric	Value
Total Data Points	20000
Training Data Points	14000
Test Data Points	6000
Model Accuracy	0.9077
AUC Score	0.9634
Training Time (seconds)	4.65

Conclusion:
By applying the Boosted Decision Tree model, I achieved high classification performance. The regularization techniques, such as shallow trees and subsampling, were crucial in preventing overfitting, while the ensemble method improved overall accuracy. The high AUC score further demonstrates the model's ability to distinguish signal events from background, making it an effective choice for classifying complex physics data with precision.

Step 6: Boosted Decision Tree Analysis on Top Variables

Purpose:

The objective of this step is to train a Gradient Boosting Classifier model using a selected subset of the most relevant variables—`numJets`, `Scalar_HT`, `jet6_pt`, `jet5_pt`, and `jet4_pt`—to classify events as signal (1) or background (0). By focusing on only the most important features, the model is expected to provide insights into how well a smaller set of variables can still perform in distinguishing between these two categories. This approach helps streamline the model without sacrificing significant performance while offering a better understanding of the contribution of each feature.

Model Configuration:

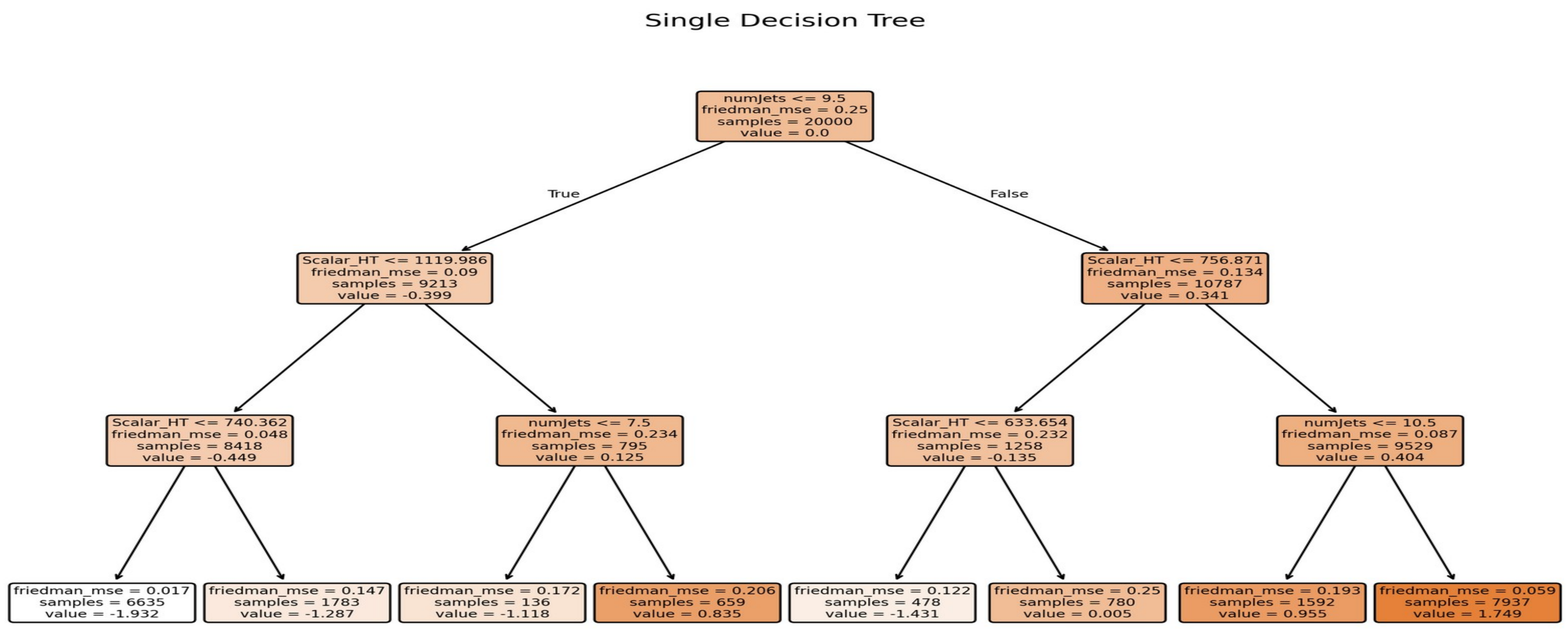
- **n_estimators=100:** A higher number of trees is used to ensure that the model has sufficient capacity to correct errors and capture the complex relationships within the data.

- **learning_rate=0.05:** The learning rate is kept small to ensure the model learns slowly and avoids overfitting by not allowing large adjustments in each iteration.

- **max_depth=3:** The decision trees are kept shallow to prevent overfitting, focusing on general patterns rather than complex, potentially noise-driven splits.

- **min_samples_split=10** and **min_samples_leaf=5:** These parameters ensure that nodes need a minimum number of samples to be split and leaves have enough samples to make stable decisions, further aiding in generalization.

- **subsample=0.8:** Random sampling of 80% of the data for each tree reduces the chance of overfitting and increases model robustness by introducing some randomness.



Performance:

Metric	Value
Total Data Points	20000
Training Data Points	14000
Test Data Points	6000
Model Accuracy	0.9057
Training Time (seconds)	2.68

Conclusion:
By focusing on a select group of the most influential variables, the Boosted Decision Tree model effectively classifies events with impressive accuracy. This analysis highlights how carefully chosen features can maintain high classification performance, offering an optimized model that avoids the complexity of using all available variables. The feature importance also provide valuable insights into which variables are most critical in distinguishing signal from background. This focused approach not only simplifies the model but also offers a more interpretable view of the data, making it a useful technique in the context of particle physics classification tasks.

4.3 Results and Conclusion:

In this analysis, we followed a structured approach to evaluate the performance of a Boosted Decision Tree model for differentiating signal events (Four Top-W Boson) from background events. The key steps involved selecting the most influential variables using a Random Forest model, training the Boosted Decision Tree model on both the full dataset and the top variables, and comparing their performance to assess the impact of feature selection on classification efficiency.

This analysis demonstrates that, even with a reduced set of key features, the Boosted Decision Tree model can still achieve near-identical performance to the full dataset. This finding underscores the importance of feature selection in enhancing the efficiency and interpretability of machine learning models without sacrificing accuracy. The results confirm that focusing on a smaller subset of important features can lead to faster, more efficient, and still highly accurate models, making it an optimal solution for real-world applications.