Aristotle University of Thessaloniki

Physics Department

MSc Computational Physics

Machine Learning Techniques for Signal-Background Discrimination in an Exotic Higgs Scenario at the LHC

Author: Konstantinos E. Stergiou

July 15, 2025

Contents

1 Introduction

The search for physics beyond the Standard Model (SM) remains a central goal of experimental efforts at the Large Hadron Collider (LHC). Supersymmetry (SUSY), one of the most prominent theoretical extensions of the SM, predicts an extended Higgs sector comprising five Higgs bosons. Among these, the presence of a heavy neutral Higgs boson decaying via exotic channels offers a rich testing ground for advanced event selection techniques.

In this study, we focus on the process $H^0 \to W^+W^-h$, where the heavy Higgs boson decays into a pair of W bosons and a Standard Model-like Higgs boson. The W bosons are assumed to decay leptonically ($W^\pm \to \ell^\pm\nu$), while the SM Higgs decays into a bottom quark pair ($h \to b\bar{b}$). This final state, characterized by two leptons, missing transverse energy, and two b-tagged jets, is challenging to isolate due to overwhelming Standard Model backgrounds.

To enhance signal-to-background discrimination, we explore and compare multiple machine learning (ML) approaches, including supervised classification algorithms trained on simulated collision data. The goal is to evaluate the performance of these models in distinguishing SUSY Higgs events from dominant background processes such as top quark pair production and diboson events. Our findings aim to contribute to the broader effort of employing ML for improved sensitivity in new physics searches at the LHC.

2 Implementation and Methodology

This section describes the tools, data preparation, and the machine learning models used to perform binary classification on the HIGGS dataset. The implementation was carried out using Python, employing widely-used libraries for data processing, machine learning, and deep learning.

2.1 Libraries Used

The following Python libraries were utilized throughout the project:

- **Pandas** and **NumPy** for data manipulation and numerical computations.
- **Matplotlib** for plotting visualizations such as ROC curves and loss graphs.
- **Scikit-learn** for preprocessing, model training (Random Forest and KNN), evaluation metrics, and cross-validation.
- **TensorFlow** and **Keras** for implementing Artificial Neural Networks (ANN).

2.2 Data Preparation

We used a cleaned subset of the HIGGS dataset, containing 8,000 examples and 28 features (21 low-level and 7 high-level engineered features). The first column indicates the binary target class (signal/background). The dataset was divided into three sets of features:

- **All features:** all 28 columns (except target)
- **Low-level features:** the first 21 features (original measurements)
- **High-level features:** the remaining 7 features (engineered)

All data were split into training and testing sets using a 75%/25% split. Feature standardization was applied using the `StandardScaler` to ensure that all models train efficiently and without scale bias.

2.3 Random Forest Classifier

The Random Forest classifier was trained using Scikit-learn's `RandomForestClassifier`. The model was tested across all three feature subsets. It was trained using default parameters, and the evaluation included accuracy, confusion matrix, ROC curves, and AUC score.

2.4 K-Nearest Neighbors (KNN)

A grid search was used to tune the KNN classifier using 5-fold cross-validation. The hyperparameters tuned included the number of neighbors (k), weighting scheme (uniform or distance-based), and distance metric (Euclidean or Manhattan). The model was trained and evaluated on all, low-level, and high-level feature subsets separately.

2.5 Artificial Neural Network (ANN)

We implemented three separate neural network models using TensorFlow/Keras:

- For **all features**: a 3-layer network (32-16-8 neurons) with `tanh` activations.
- For **low-level features**: the same architecture with adjusted batch size and training epochs.
- For **high-level features**: a deeper network (128-64-32-16-8 neurons) to compensate for the smaller input dimensionality.

All networks used binary crossentropy loss and the Adam optimizer. The training process was visualized via loss curves, and performance was measured using accuracy, confusion matrices, ROC curves, and AUC scores.