Quantum Computing - Machine Learning Project

Georgia Pavlina Tsoukaneri

June 2025

1 Introduction

The subject of this report is the classification of exotic Higgs decays using Machine Learning. The vast majority of particle collisions do not produce exotic particles. It is thus crucial to be able to recognise collisions which produce particles of interest (signal) from those producing other particles (background).
In order to classify the events as signal or backround we will use a combination of different features. These features are distinguished in low-level features, which are raw output from the accelerators and high-level features, which are quantities derived from the low-level features.

2 Methodology

We will be using three different machine learning methods to classify our data: **K Nearest Neighbour, Random Forest, Artificial Neural Network**.
In order to implement these methods, we will make use of the Python libraries sklearn (for knn and random forest) and tensorflow (for the neural networks). We will also be using pandas to access and clean our data.

Data cleaning and preparation

After importing all of our data to a pandas dataframe, we checked the datatype of each column to verify that we only had clean numerical data. Only one of the values for the features seemed to have been mistyped, being written as 0.000000000000000000e+00.1 instead of 0.0. This resulted in the pandas library being unable to interpret the scientific notation and storing the value as a string. We made sure to replace it with the correct numerical value.

Splitting Dataset for training and testing

All supervised machine learning methods require the dataset to be split in two parts: one of them is used to train the model, whereas the other one is used to verify the accuracy of the model. We decided to use a 70%- 30% percent split for our dataset.
Since we want to make separate classifications based on the low-level and high-level features we also split the training sets in two parts: one of them using the low level features (columns 1-21) and the other the high level features (columns 22-28).

Training set feature scaling

Distance based models like knn and random forest are only useful if the values of the features used are scaled. We will standardize the data in our training set using a standard scaler. This will substract the mean from every value and then divide with the standard deviation, creating a dataset where all features have mean 0 and std 1. It is very important to fit test data using the same scaler as the training data.

Neural Network parameters

While designing the neural network, we decided to opt for a shallow network with 2 hidden layers, to prevent overfitting. This is crucial for our small dataset. The parameters used were the following:

- **Architecture**
  - Input layer: 21 or 6 neurons (for low and high level feature models respectively)
  - Hidden layer 1: 24 neurons (ReLU activation)

  - Hidden layer 2: 12 neurons (ReLU activation)
  - Output layer: 1 neuron (Sigmoid activation)

- **Training Configuration**
  - Optimizer: Adam (adaptive learning rate). We chose this optimizer because it is a good choice for beginners and offers flexibility without needing any fine tuning.
  - Loss function: Binary cross-entropy
  - Metric: Accuracy

- **Activation Functions**
  - ReLU for efficient learning in hidden layers
  - Sigmoid for binary probability outputs

- **Training Protocol**
  - Batch size: **32**. 0.6% of the data per update (175 batches/epoch). This is reasonable for a small dataset, balancing avoiding local minima and minimizing noise. It also is memory efficient.
  - Epochs: **50**. Common choice for a small dataset. Chosen through trial and error. Epoch sizes beyond this number didn't seem to substantially improve the accuracy of the model.

3 Results and Analysis

Confusion matrices and accuracy metrics

To present our results we will be using confusion matrices for visualization. A confusion matrix is a mapping of the predicted values versus the true values of events in the testing set. It effectively takes the format:

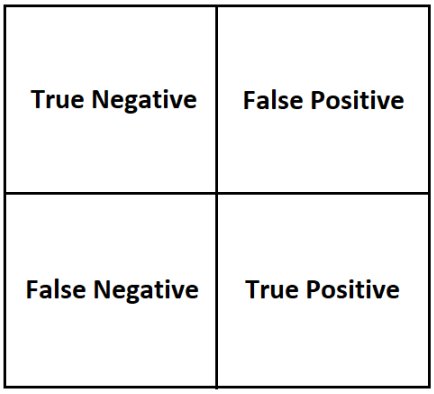

Figure 1: Confusion matrix of a classifier

To better quantify these results we would like to use the precision and recall metrics:

- Precision: Ability to identify **only** true signals or percentage of detections that are actually signals. Calculated as $\frac{TP}{TP+FP}$.
- Recall or sensitivity: Ability to identify **all** true signals or percentage of true signals that end up being detected . Calculated as $\frac{TP}{TP+FN}$.

While also use the accuracy metric:

- Accuracy: Percentage of sample identified correctly. Calculated as $\frac{TP+TN}{totalsamples}$. While clearly an important metric, we cannot limit our analysis to that, as it can be confusing in imbalanced datasets. For example, in a set where 95% of the detections are background, a classifier that always predicts "backround" would have an accuracy of 0.95 percent.

K-Nearest Neighbours

The accuracy of the knn method is visualized through the confusion matrix below.

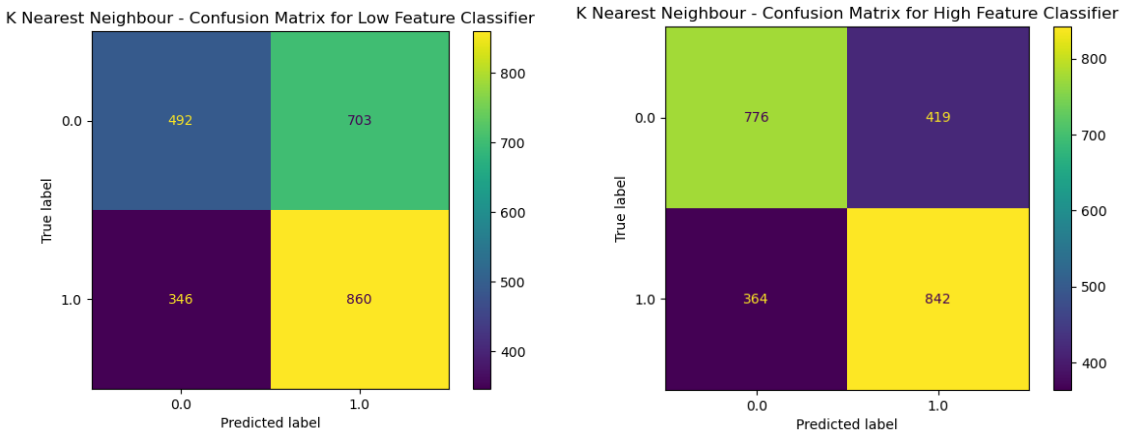

Figure 2: Confusion matrix for knn classifier. Left: Low-level features, Right: High-level features.

We also present results for the precision and recall metrics:

Table 1: Precision and Recall Comparison for Low-Level vs. High-Level Features

Metric	Low-Level Features	High-Level Features
Precision	0.550	0.668
Recall	0.713	0.698

The high-level features demonstrate superior precision (0.668 vs 0.550) while maintaining comparable recall (0.698 vs 0.713), indicating they better capture the physical signatures of exotic decays.

Random Forest

The accuracy of the random forest method is visualized through the confusion matrix below.

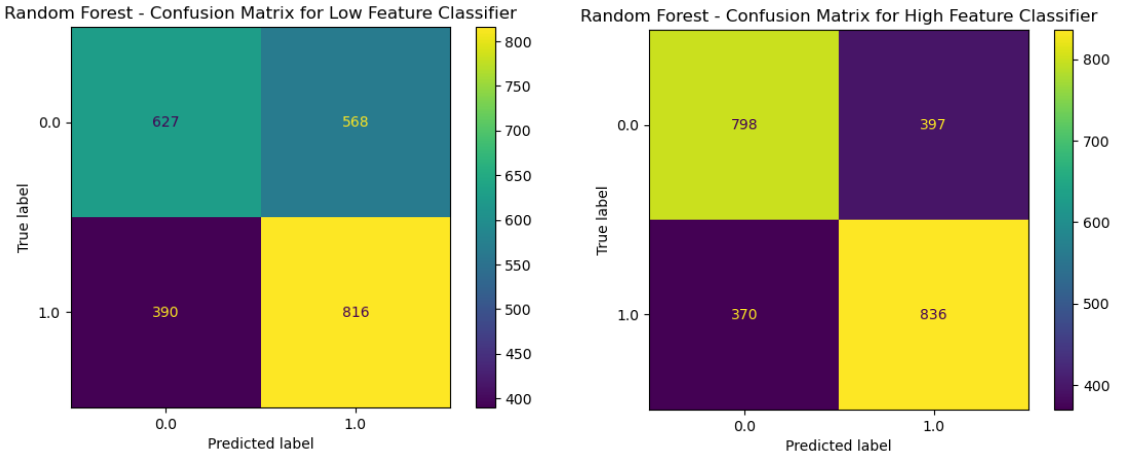

Figure 3: Confusion matrix for rf classifier. Left: Low-level features, Right: High-level features.

We also present results for the precision and recall metrics:

Again, the high-level features demonstrate superior precision (0.678 vs 0.590) and comparable recall (0.693 vs 0.077).

Table 2: Precision and Recall Comparison for Low-Level vs. High-Level Features

Metric	Low-Level Features	High-Level Features
Precision	0.590	0.678
Recall	0.677	0.693

Artificial Neural Network

The accuracy of the neural network method is visualized through the confusion matrix below.

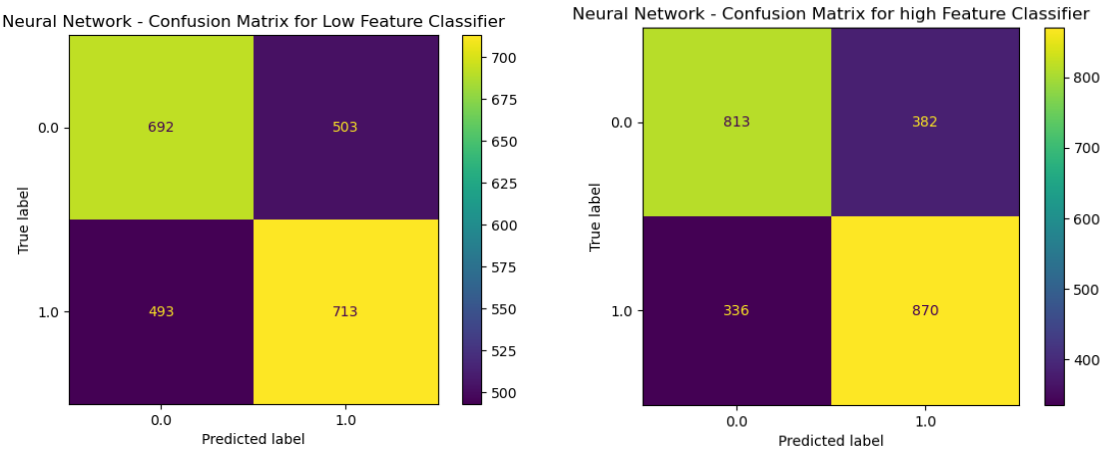

Figure 4: Confusion matrix for neural network classifier. Left: Low-level features, Right: High-level features.

We also present results for the precision and recall metrics:

Table 3: Precision and Recall Comparison for Low-Level vs. High-Level Features

Metric	Low-Level Features	High-Level Features
Precision	0.568	0.695
Recall	0.591	0.713

The analysis based on low level features seems to underperform in comparison to the analysis of low-level features of other methods. The analysis for high level features though hugely improves the results, producing the best overall results both in regards to precision and recall.

Comparative analysis of the accuracy of the three methods.

Table 4: Accuracy Score Comparison for Different Methods

Method	Low-Level Features	High-Level Features
k-Nearest Neighbour	0.563	0.674
Random Forest	0.601	0.681
Artificial Neural Network	0.585	0.701

Based on the accuracy metric, high-level features massively outperform the low-level features when used to train the classifiers.

4 Discussion

Taking the results and analysis into account we come to the following conclusions:

- High level features consistently outperform low level feature methods, as they take physical knowledge into account, creating "physically informed" classifier methods.

- The neural network has the highest recall, precision and accuracy than all methods.
- For high-level features, precision improved comparatively more than recall. This is particularly important for the extinction of false positives and the subsequent analysis of only true signals.
- Further analysis could expand on how a larger dataset affects the performance metrics and whether low-level feature analysis catches up with high-level feature analysis.

5 References and Resources

- The analysis follows the paper "Searching for exotic particles in high-energy physics with deep learning' by P. Baldi, P. Sadowski and D. Whiteson.
- The website Geeks for Geeks was used for conceptual understanding of the classifier methods and the definitions of the accuracy metrics.
- All calculations where carried out using local resources using the Python libraries scikkit-learn and tensorflow.