Aristotle University of Thessaloniki
Physics Department
MSc Computational Physics

Final Project 2:
Detection of hyper-symmetric Higgs bosons

Testing machine learning and artificial neural network classification algorithms

Subject: Computational Quantum Physics
Implemented by: Ioannis Stergakis
AEM: 4439

July 2024

Contents

List of Figures

1 Introduction

Both machine learning (`ML`) and artificial neural networks (`ANN`) are very useful tools for data analysis. We can use them to change the scale of our data to make it more efficient for our calculations and analysis, or even to identify and fill in missing data in our data frames. Most importantly, however, they provide us with the ability to train models for classifying and predicting the known (or unknown) outcome from a set of observable variables.

In this assignment, we will work with supervised machine learning and artificial neural network models to optimize event selection in the search for hyper-symmetric Higgs bosons at the LHC. Specifically, we will test the effectiveness of two (`ML`) algorithms, the `SVM` (Support Vector Machine) and `RF` (Random Forest) algorithms, and of the `TensorFlow` network in separating signal and background events from a given data set.

2 Methodology

2.1 Data imputation and feature scaling

The data we have to analyze are included in the `HIGGS_8K.csv` file. Our work is presented in the `FinalProject2_Higgs_ML.ipynb` notebook, for machine learning classification and in the `FinalProject2_Higgs_ANNs.ipynb` notebook for the `TensorFlow` neural network classification, respectively. In both notebooks, we import the data using the `pandas` module and perform a preprocessing sequence.

This includes the imputation and filling of the missing data, as well as the splitting and scaling of the data. First, we detect that the data in the 17th column are in `string` format and will effect the analysis. Thus, we typecast them to `float` format and continue. We detect missing data (i.e 0s) in columns 9, 13, 17 and 21 and replace them with `NaN`. Then we replace those `NaN`s with the mean value of the respective column, using the `fillna` command.

The splitting of the data comes next. First, we separate the outcome (i.e the first column) from the data and the explanatory data into low level (i.e columns 1-21) and high level (i.e columns 22-28) quantities. We keep the whole explanatory data (i.e columns 1-28), too. Then, we split each of the aforementioned, into train and test sets using the `train_test_split` command of the `sklearn.model_selection` package. We choose the 75% of the data to be the train set and the rest 25% the test set.

2.2 Classification using machine learning (ML) algorithms

Proceeding to the machine learning classification, in section 2.1 of the notebook we perform a `grid search` combined with `k-fold` with **6** folds (splits) to determine the best `SVM` model for each of the low level, high level and whole explanatory quantities. We change two hyperparameters of the `SVM` algorithm: the `C` value and the `kernel`, by giving them the choices **[0.1, 0.25, 0.5, 0.75, 1.0, 1.25, 1.5, 1.75, 2.0]** and **['linear', 'poly', 'rbf', 'sigmoid']**, respectively.

We perform a similar grid search in section 3.1 of the notebook, for the Random Forest (`RF`) algorithm. This time we change the `n_estimators` and `criterion` hyperparameters, giving them the choices **[10, 50, 100, 200]** and **["entropy","gini","log_loss"]**, respectively.

2.3 Classification using the `TensorFlow` neural network (NN)

On the other hand, in `FinalProject2_Higgs_ANNs.ipynb` notebook, using the `TensorFlow` and `keras` modules we built a neural network (`NN`) with input layer, 1 hidden layer and output layer and a deep network (`DN`) with input layer, 2 hidden layers and output layer, for each of the quantities category.

As of the `NN` for the whole of the explanatory quantities, the input layer has **28** neurons, the hidden layer **20** and the output layer **1** neuron. On the other hand, the `DN` has **28** neurons on the input layer, **20** in the 2 hidden layers and **1** in the output layer.

For the low level quantities, the `NN` has **21** neurons in the input layer, **30** in the hidden layer and **1** in the output layer. The `DN` has **21** neurons in the input layer, **30** in the 2 hidden layers and **1** in the output layer.

Finally, for the high level quantities, the `NN` has **7** neurons in the input layer, **12** in the hidden layer and **1** in the output layer. The `DN` has **7** neurons in the input layer, **12** in the 2 hidden layers and **1** in the output layer.

We use the `relu` activation function for the input and hidden layers and the `sigmoid` activation function for the output layer, along with the **1** neuron in it, since we have a two-categories (signal-background) classification. At the compilation of the network, we choose `adam` as the optimizer, `binary_crossentropy` as the loss function, and include metrics for `accuracy`. For the training of the models, we choose a `batch_size` of **100** and **150** **`epochs`**, in order to avoid overfitting. For more about, the classification utilizing `ANN`s, in hyper-symmetric Higgs senarios see [1].

3 Results

3.1 `SVM` and `RF` classification

The grid search for the `SVM` algorithm resulted to the value $C = 2$ and `'rbf' kernel`, as the best model's hyperparameters for all explanatory quantities. For both the low and high level, quantities it resulted to the value $C = 1.75$ and `'rbf' kernel`. The accuracy of the best `SVM` model for all explanatory variables is **0.657671164417791**, whereas of the best `SVM` model for the low level quantities is **0.5767116441779111** and the one for the high level quantities is **0.6756621689155422**. We also have the following confusion matrices

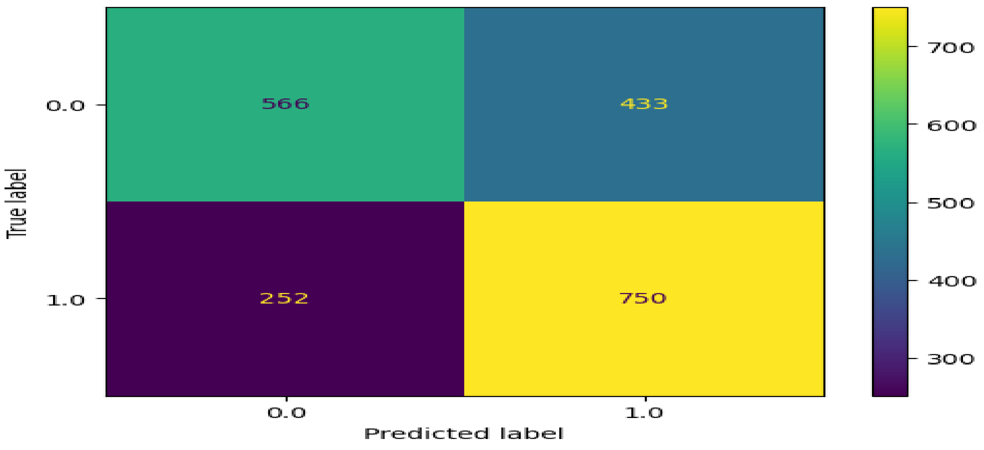

Figure 1: Confusion matrix of the best `SVM` model for all quantities

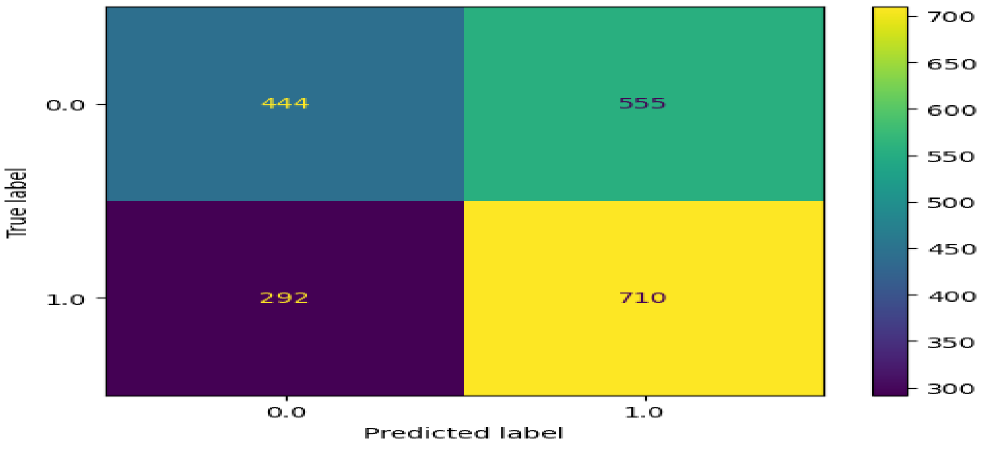

Figure 2: Confusion matrix of the best `SVM` model for low level quantities

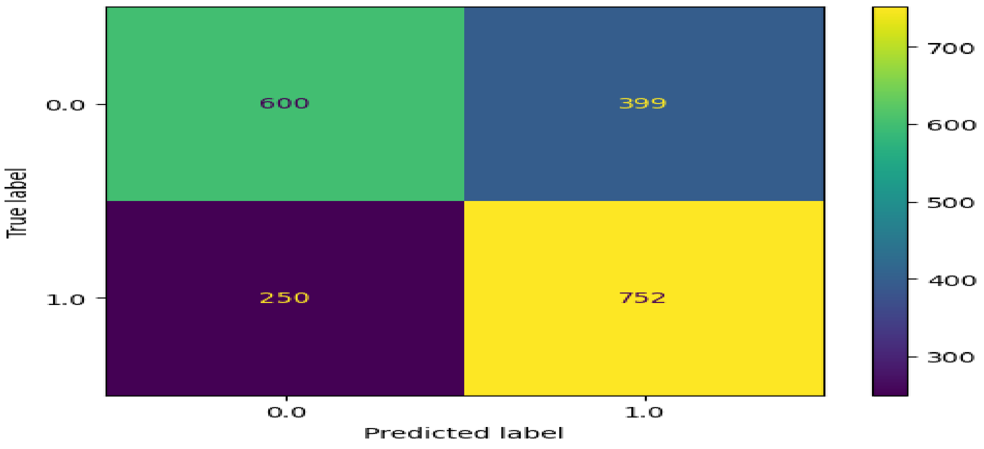

Figure 3: Confusion matrix of the best `SVM` model for high level quantities

and the following ROC curve

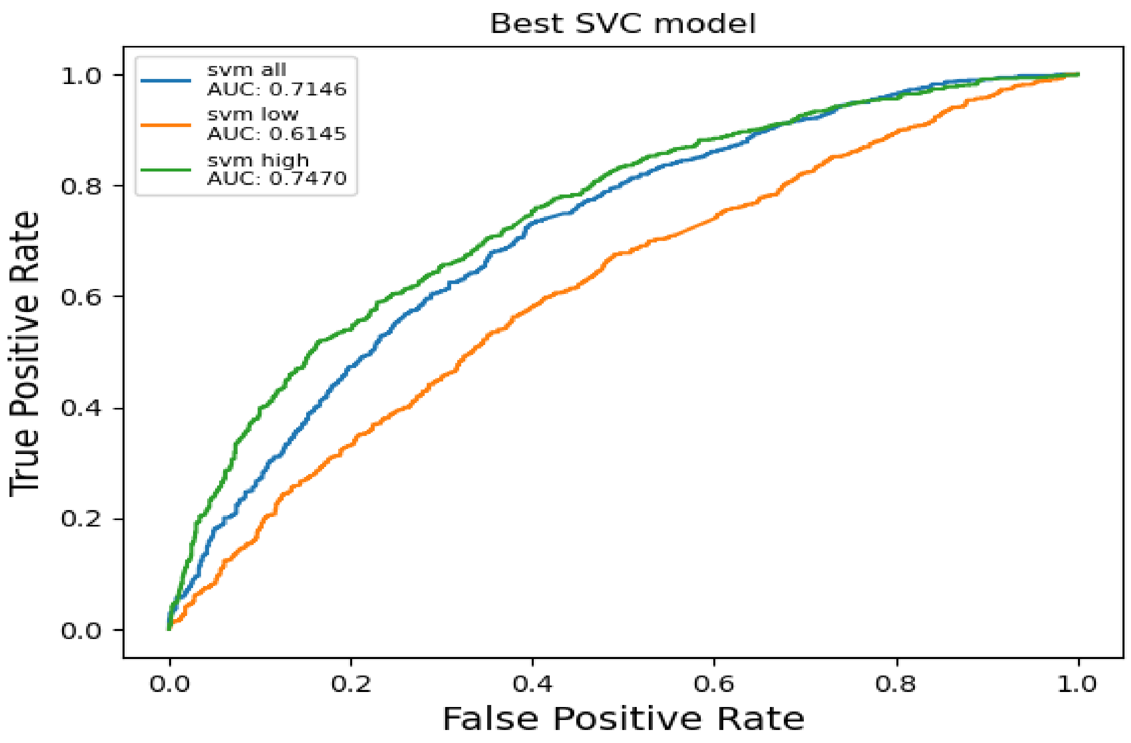

Figure 4: ROC curve of the best SVM models

We can see that the best SVM model is the one for the high level quantities, with AUC score **0.7470**, and the worst the one for the low level quantities, with AUC score **0.6145**.

Regarding, now, the grid search for the RF algorithm, this results to the 'entropy' criterion and 200 'n_estimators', as the best model's hyperparameters for all explanatory quantities. For both the low and high level, quantities it results to the 'log_loss' criterion and 200 'n_estimators'. The best RF model for all explanatory variables has accuracy **0.69665167416291**. The best RF model for the low level quantities has **0.57621189405297** accuracy and the one for the high level quantities has **0.680159920039** accuracy. We also have the confusion matrices and the respective ROC curve in the following pages. It is clear from Figure 8 that the best RF model is this time, the one for all explanatory quantities, with AUC score **0.7856** and the worst, once again, for the low level quantities with AUC score **0.6332**. However, the high level quantities model is very close to the one of all explanatory quantities with AUC score **0.7670**.

Therefore, we can make a first assumption for the behavior of the data. The high level quantities, suffice to classify the outcome between signal and background and in cases they might outperform the all explanatory variables models. On the other, hand the low level may consist mostly of noise or even quantities that are related to one another or even to the high level quantities, hence they add unnecessary information to the models. Thus, it is optimal to limit their use in the classification process.

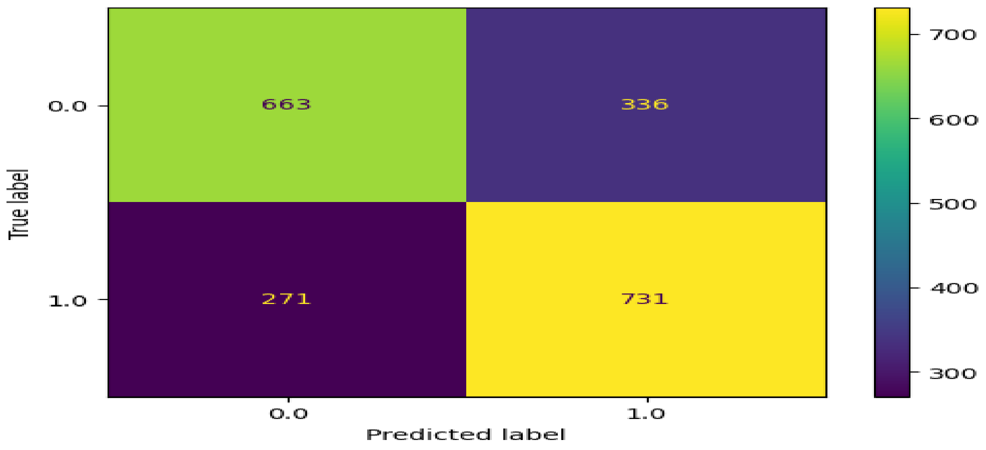

Figure 5: Confusion matrix of the best `RF` model for all quantities

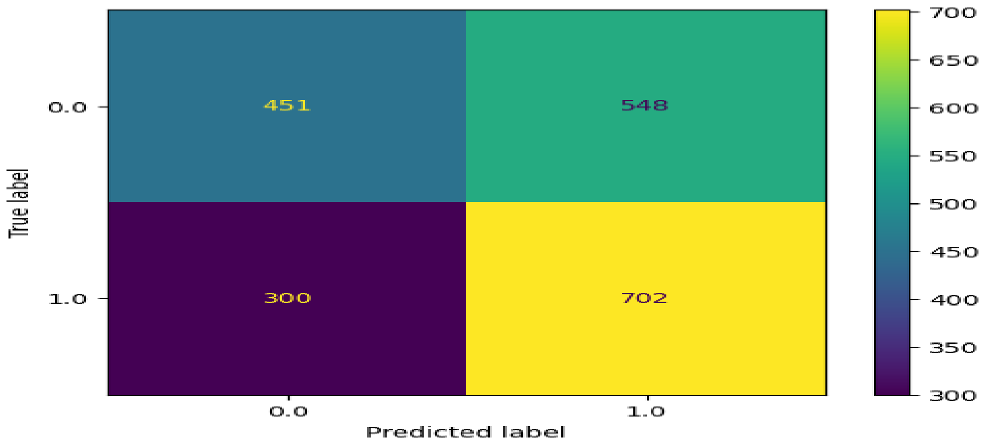

Figure 6: Confusion matrix of the best `RF` model for low level quantities

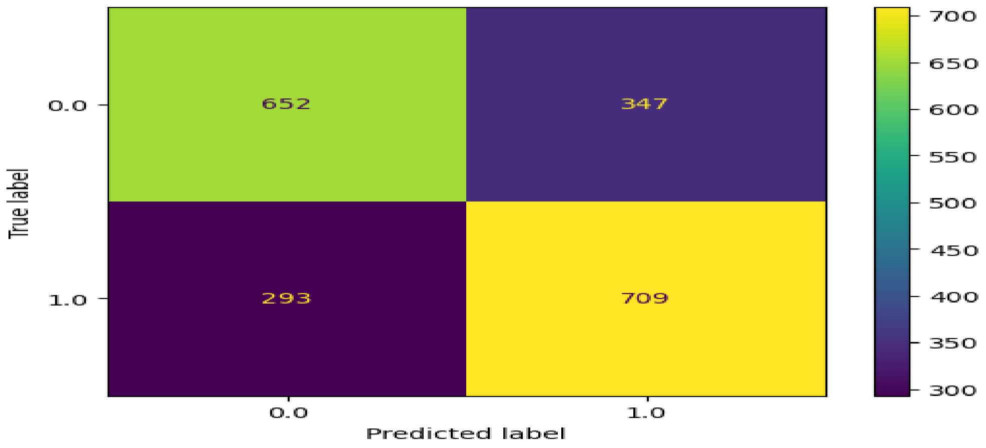

Figure 7: Confusion matrix of the best `RF` model for high level quantities

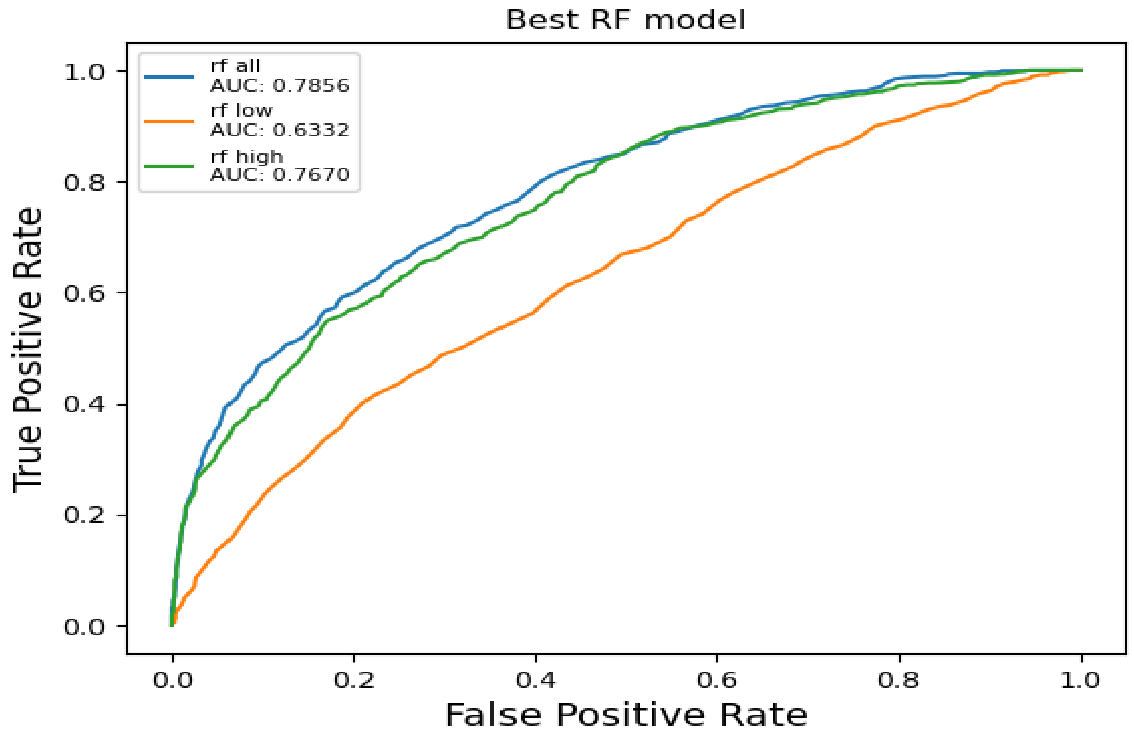

Figure 8: `ROC` curve of the best `RF` models

3.2 `TensorFlow` classification

Let's see now, if the `TensorFlow` classification confirms our `ML` result. For all explanatory models, the `NN` model has **Test loss: 0.63028** and **Test accuracy: 0.66416**. On the contrary, the `DN` has **Test loss: 0.73010** and **Test accuracy: 0.65217**. For the low level quantities the `NN` has **Test loss: 0.71366** and **Test accuracy: 0.55522**, while the `DN` model has **Test loss: 0.88356** and **Test accuracy: 0.54022**. Finally, for the high level quantities the `NN` has **Test loss: 0.58916** and **Test accuracy: 0.67166** and the `DN` has **Test loss: 0.56785** and **Test accuracy: 0.69065**.

So, we can see the `DN` model is not always better than the simpler `NN` model. In fact, only for the high level quantities the `DN` outperforms the `NN`, i.e having better accuracy and smaller loss. This, can be, also confirmed from the `ROC` curve in Figure 9. Beyond, that the high level `NN` and `DN` are the best models (but not with much difference from the all quantities models), while the worst are once again the models for the low level quantities. This validates our assumptions, of the negative contribution the low level quantities have to the classification process.

Last but not least, we have to mention that all the results above have a factor of randomness in them, since lots of `ML` and `ANNs` procedures use random seeds during their operation. Therefore, one might result in slightly different outcomes when compiliing our codes in the attached `FinalProject2_Higgs_ML.ipynb` and `FinalProject2_Higgs_ANNs.ipynb` notebooks. However, we expect the same general behavior in the classifcation results, i.e a minor

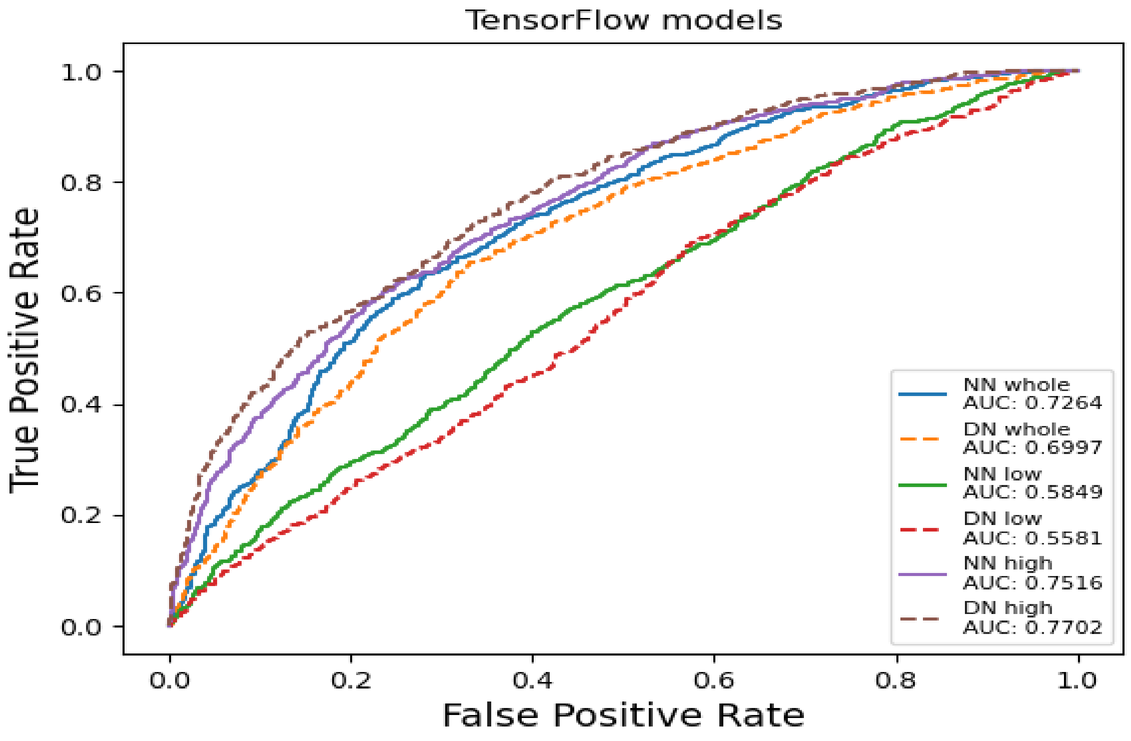

Figure 9: `ROC` curve of the `TensorFlow` models

difference between the on average good results for high level and all quantities, and the failure of the low level quantities to produce safe and reliable results.

References

[1] Pierre Baldi, Peter Sadowski, and Daniel Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nature communications*, 5(1):4308, 2014.