Using Random Forest Classifier for particle identification in the ALICE Experiment

Tomasz Trzciński[1], Łukasz Kamil Graczykowski[2] and Michał Glinka[1]
for the ALICE Collaboration

[1]Institute of Computer Science, [2]Faculty of Physics
Warsaw University of Technology, Poland,
t.trzcinski@ii.pw.edu.pl, lukasz.graczykowski@pw.edu.pl,
mglinka2@stud.elka.pw.edu.pl

Abstract. Particle identification is very often crucial for providing high quality results in high-energy physics experiments. A proper selection of an accurate subset of particle tracks containing particles of interest for a given analysis requires filtering out the data using appropriate threshold parameters. Those parameters are typically chosen sub-optimally by using the so-called "cuts" – sets of simple linear classifiers that are based on well-known physical parameters of registered tracks. Those classifiers are fast, but they are not robust to various conditions which can often result in lower accuracy, efficiency, or purity in identifying the appropriate particles. Our results show that by using more track parameters than in the standard way, we can create classifiers based on machine learning algorithms that are able to discriminate much more particles correctly while reducing traditional method's error rates. More precisely, we see that by using a standard Random Forest method our approach can already surpass classical methods of cutting tracks.

Keywords: Monte Carlo tracks, Random Forest classification

1 Introduction

Particle identification (PID), *i.e.* the identification of the mass and flavour composition of particles produced during a collision, is very often a first preprocessing step of a typical analysis in high energy physics experiments. It is of a particular importance in the case of ALICE (A Large Ion Collider Experiment) [1], one of the experiments of the Large Hadron Collider (LHC) [2], whose goal is to study all aspects of ultra-relativistic heavy-ion collisions (lead–lead (Pb–Pb)) in order to measure the properties of the Quark-Gluon Plasma [3, 4]. ALICE is a complex detector composed of 18 different detection systems that employ various methods of interaction with highly energetic particles to measure physical phenomena.Thanks to the complexity of detection systems, a separation of signals left by different particle species is possible in a very broad momentum range, from around 100 MeV/c up to around 10 GeV/c, making ALICE the most powerful of the LHC experiments in terms of PID capabilities.

PID requires filtering out a signal corresponding to a given particle species from signals produced by other particle types. This is typically achieved by applying certain threshold parameters on the deviation of the signal in a given detector from its nominal value. Those parameters are typically chosen by using the so-called "cuts" – sets of simple linear classifiers which remove unwanted data below or above a given threshold value. This method works well when the signals are well separated from each other and information from one detector is sufficient. Yet, when they start to overlap and proper combination of the PID data from two or more detectors is required, setting correct threshold values becomes unintuitive and non-trivial. The sub-optimality of such approach typically results in low accuracy, efficiency, or purity of the sample of selected particles, hence reducing available data and reducing statistical power of physical analyses.

In this paper, we address the above-mentioned shortcoming of the currently used method and we propose to improve the selection of particles using machine learning methods. More precisely, we propose a set of classifiers, called the Random Forest [5], that are trained specifically to improve the discrimination between particle types. The classifiers improve the sensitivity, defined as a ratio of correctly classified signal particles to its total amount, over the currently employed method, while not decreasing the precision of the method. In this paper, we show how a state-of-the-art Random Forest classifier can be used to solve the address the problem of particle identification. We evaluate the quality of the proposed selection method and compare it with the traditional method, proving that our approach can significantly outperform the currently used algorithm both in terms of the number of correctly classified particles and the error rates of their selection.

2 Particle identification with TPC and TOF

PID of light-flavour charged hadrons (pions, kaons, protons), the most abundantly produced particles in heavy-ion collisions, is usually performed using two ALICE detectors, that is the Time Projection Chamber (TPC) [6] and the Time-Of-Flight system (TOF) [7]. Understanding the detector response of both of them is crucial for any PID technique.

The TPC – the main tracking device of ALICE – is a cylindrical gaseous detector which provides, for a given particle, measurements of the mean energy loss per unit path length, $\langle \mathrm{d}E/\mathrm{d}x \rangle$, from up to 159 independent $\mathrm{d}E/\mathrm{d}x$ measurements along its trajectory. The $\langle \mathrm{d}E/\mathrm{d}x \rangle$ as a function of particle's momentum p is described by the Bethe-Bloch empirical formula, whose parameters depend on various factors, including the detector intrinsic resolution, track reconstruction details, parameters of the gas, and others [8]. The distribution of measured $\langle \mathrm{d}E/\mathrm{d}x \rangle$ around the value expected from the Bethe-Bloch parameterisation has a Gaussian profile with a standard deviation σ_{TPC}. Therefore, the best PID estimator for the TPC detector is defined as a distance, in numbers of standard deviations $N_{\sigma_{\mathrm{TPC}}}$, of the measured $\langle \mathrm{d}E/\mathrm{d}x \rangle$ to the nominal signal from the Bethe-Bloch curve.

The TOF detector provides the measurements of velocity of a charged particle by measuring its time of flight t_{TOF} over a given distance along the particle trajectory l. The arrival time is measured by employing the Multigap Resistive Plate Chamber technology which have an intrinsic resolution of 80 ps. For each particle type the measured arrival time, t_{TOF}, is compared to the nominal (expected) time, t_{exp}. The former is defined as a difference between the arrival time in TOF and the event collision time, evaluated using a sophisticated procedure (for details see [9]), while the latter is the time it would take for a particle of a given mass to travel from the interaction point to the TOF. The distribution of the arrival time measured in TOF around the expected time has a Gaussian profile with a standard deviation $\sigma^{\mathrm{TOF}}_{\mathrm{PID}}$. By analogy, the best PID estimator for the TOF detector is defined as a distance, in numbers of standard deviations $N_{\sigma^{\mathrm{TOF}}_{\mathrm{PID}}}$, of the measured arrival time t_{TOF} to the nominal signal t_{exp}. In practice, t_{TOF} is often expressed as $\beta = v/c$, where $v = l/t_{\mathrm{TOF}}$ is the particle's velocity and c is the speed of light.

Figure 1 shows typical TPC energy loss and TOF velocity measurements as a function of particle's momentum from proton–proton collisions at the center-of-mass energy of $\sqrt{s} = 13$ TeV measured by ALICE.

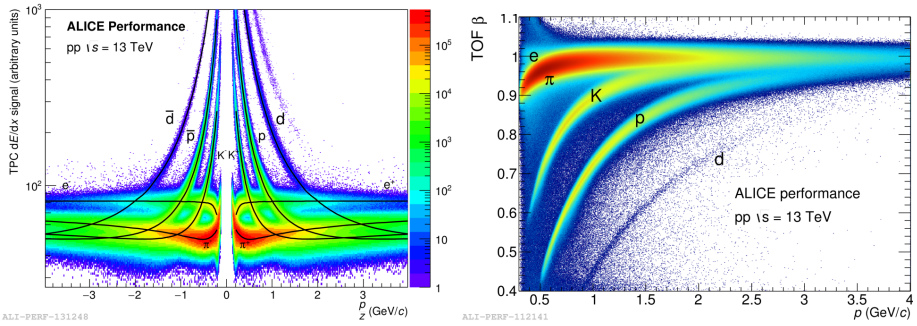

Fig. 1. Results of the measurements of (left) the energy loss in the TPC with lines corresponding to Bethe-Bloch parameterisation and (right) particle's velocity β in the TOF detector as a function of particle's momentum.

3 Baseline PID method

The typical approach to the PID involves applying simple linear selection criteria where single cut-off values are used to accept or reject tracks with specific $N\sigma^{\mathrm{TPC}}_{\mathrm{PID}}$ and $N\sigma^{\mathrm{TOF}}_{\mathrm{PID}}$ based on particle's momentum. They are often chosen arbitrarily depending on the goals of the analysis (i.e. whether maintaining the very high purity of the selected sample is required) and experience of a scientist performing the study. In addition, other parameters of the track which may be relevant for

PID in a non-trivial way are omitted. Finally, adjusting the cut-off values in order to achieve the desired parameters of the sample requires a lot of time and effort on the physicists side.

Some alternative approaches for PID exist, *e.g.* the Bayesian PID [10], which relies on Bayesian probabilistic approach to the problem of particle identification. Although this method was proven to provide higher signal-to-noise rations, it is not widely used in practice yet and therefore in this work we use the traditional PID as a baseline for our method.

4 Our approach

In this paper, we propose an approach for machine learning-based PID that is based on the Random Tree method [11]. More precisely, we choose a classifier, called the Random Forest [5], that uses numerous random decision trees to classify given observation. In this section, we describe the method and justify its selection for our application.

4.1 Random Forest algorithm

Random Forest is an ensemble of decision trees that generate the classification decision based on a set of sub-decisions. In each decision tree, attributes are represented as nodes and classifier's decisions as leaves. Each node is created by selecting best attribute from the fixed size random subset of attributes used to train current tree. Their quality might be assessed via calculating entropy gain for each attribute or its Gini index, which is defined as probability of wrong classification while using only a given attribute. Each node splits a dataset into two subsets, trying to maximize the chances that the samples of the same class end up in the same subset. If one of those subsets has low number of data samples from other classes, it is converted into a leaf and no further splitting is needed. Minimal impurity of a node needed for this to happen equals zero by default, but can be modified as one of classifier's hyper-parameters. The whole process is then repeated to create new nodes. In a classical approach, Random Forest trees are not pruned in any way and those steps are repeated until all leaves are created.

The Random Forest classifier has several hyper-parameters that can be tuned:

- number of decision trees inside the forest
- maximum depth of each decision tree
- minimal impurity of a node for it to be converted into a leaf
- maximum number of attributes used per tree training
- minimal impurity decrease of resulting subdatasets for a node to be created

The final classifier is created by training multiple decision trees. When final classification is to be made, each of the tree processes the data independently. Then, the final decision is chosen as a result of majority voting from all trees inside the forest.

4.2 Unique properties of Random Forest

In the context of PID, processing high amounts of data at large speed is essential. This is why the Random Forest classifier is a perfect fit. Thanks to its simplicity and ability to scale horizontally[1], the classification decision process can be sped up not only by increasing the computational power, but also by using more machines. Many independent training processes on separate datasets can be parallelized on separate machines and later all resulting decision trees can be simply aggregated into a single classifier. This approach enables to fully exploit the potential of GRID [12], a global collaboration of more than 170 computing centres in 42 countries providing global computing resources to store, distribute and analyse enormous amounts of data as part of CERN's computing infrastructure.

Additionally, the Random Forest classifier is very resistant to *overfitting*. Overfitting is a common phenomenon known in machine learning which defines the situation when the classifier loses its ability to correctly classify samples outside of a training dataset, because its parameters are over-tuned on the training dataset.

Another advantage of the Random Forest over other machine learning methods, such as Support Vector Machines, is its interpretability. The analysis of the parameters used for a given split is straightforward and does not require any additional tools. Thanks to this property, we can create a simple set of 'sanity checks' to see whether particle classification is based on a complimentary set of attribute choices rather than some hidden correlations within our dataset.

Finally, the Random Forest classifier is used in numerous applications across domains, *e.g.* genetics – to identify DNA binding proteins [13] – or medicine – to classify diabetic retinopathy [14]. This wide adoption of the Random Forest method indicates the potential of this method and leads to a significant amount of resources, such as tutorials, publications and libraries, that can be found in the Internet and used to improve the final performance.

4.3 Implementation

In our work, we use a Python implementation of the Random Forest [5] classifier provided by scikit-learn package [15], instead of the ROOT environment, which is widely used at CERN and it is in fact our ultimate production environment.

We use Python for development purposes, as it offers a diverse set of tools used in both machine learning and data engineerings tasks. Thanks to C and Fortran snippets of code integrated in the Python backend, all calculations are efficient while a high level of API abstraction enables relatively easy code development. As we envision ROOT as our final production envrionment, we maintain a full integration between those two distinct systems by using external libaries

[1] Horizontal scalability is an attribute of a system, which may be expanded by adding new nodes (machines, servers, computers), rather than by only increasing computing power of existing ones.

that allow us to convert CERN datatypes into most commonly used Python one – pandas DataFrame [16]. After the conversion is done we can use Python tools not only to train a classifier, but also to evaluate its performance and tune its parameters.

5 Experiments

In this section, we describe the results of a comparison between our approach and the traditional PID method. Both methods are used to perform a binary classification with kaon particles labeled as a signal and all other non-kaon particles as a background. We choose kaons, as they are the most abundantly produced particles in a typical pp collisions and, therefore, an excellent use case for our PID algorithm.

The remainder of this section is organized as follows. We first present the dataset generated using Monte Carlo simulations that we use in our evaluation. We then present the results of the initial experiments that aimed at determining the importance of input parameters for the final classification task. We then outline the final results of our evaluation which showsbb that the proposed Random Forest classifier significantly outperforms currently used PID methods in the binary discrimination task of the kaon particle classification.

5.1 Dataset

As our dataset, we use Monte Carlo proton–proton data at the center of mass energy of $\sqrt{s} = 7$ TeV, generated by PYTHIA 6.4 [17] model, Perugia-0 [18] tune. After generating the particles,we transport (process) them using GEANT3 [19] package, simulating the passage of the particles through the detector medium. We also perform a full simulation of the detector response as well as the reconstruction of full trajectories of the generated particles. The experimental conditions correspond to 2010 data-taking period. This way, we obtain over 269,272 particle tracks with the associated label determining its particle class, in our case kaon vs non-kaon. This information is crucial for classifier training as we can then cast the PID as a classification problem and assess the performance of the Random Forest classifier when solving it.

For completeness, we present here the exact criteria used to select particle trajectories that form our dataset:

- for tracks with $p_{\mathrm{T}} > 0.5$ GeV/c the combined information from both the TPC and TOF was used, ${N^K_{\sigma,\mathrm{PID}}}^2 = {N^K_{\sigma,\mathrm{TPC}}}^2 + {N^K_{\sigma,\mathrm{TOF}}}^2$, resulting in a circular cut in the $N^K_{\sigma,\mathrm{TPC}}$ and $N^a_{\sigma,\mathrm{TOF}}$ space,
- for tracks with $p_{\mathrm{T}} < 0.5$ GeV/c, where only a few have an associated signal in the TOF and information only from the TPC was used $N^K_{\sigma,\mathrm{PID}} = N^K_{\sigma,\mathrm{TPC}}$.

For each of the particle trajectories, our dataset contains 36 attributes that can be used as an input of the classification method:

- $\beta = v/c$ – particle's velocity measured in TOF relative to the speed of light c,
- p_{x}, $p_{\mathrm{y}}, p_{\mathrm{z}}$ – components of the particle's momentum along x, y, z directions, respectively,
- $p = \sqrt{p_{\mathrm{x}}^2 + p_{\mathrm{y}}^2 + p_{\mathrm{z}}^2}$ – total momentum of a particle,
- No. of TPC clusters – number of TPC clusters belonging to a given particle track,
- TPC signal – mean energy loss signal measured the TPC detector,
- $N_{\sigma^{\pi}_{\mathrm{TPC}}}$, $N_{\sigma^{\mathrm{K}}_{\mathrm{TPC}}}$, $N_{\sigma^{\mathrm{p}}_{\mathrm{TPC}}}$, $N_{\sigma^{\mathrm{e}}_{\mathrm{TPC}}}$ – difference expressed in units of standard deviation σ between the measured and the expected signals for a pion, kaon, proton and electron from the TPC detector, respectively,
- $N_{\sigma^{\pi}_{\mathrm{TOF}}}$, $N_{\sigma^{\mathrm{K}}_{\mathrm{TOF}}}$, $N_{\sigma^{\mathrm{p}}_{\mathrm{TOF}}}$, $N_{\sigma^{\mathrm{e}}_{\mathrm{TOF}}}$ – difference expressed in units of standard deviation σ between the measured and the expected signals for a pion, kaon, proton and electron in the TOF detector, respectively
- cov0-20 – components of a covariance matrix between x,y,z spatial coordinates and the components of the particle's momentum along x,y,z directions.

5.2 Attribute importance

To understand the impact of each of the 36 attributes on the final classification results, we train our Random Forest classifier using only subsets of attributes, *i.e.* exluding the attribute whose importance we evaluate. We then measure the deterioration of the classifier performance by calculating the difference in purity (recall) between the classifier trained with a full set of attributes and without the evaluated one. We call this metric *mean impurity decrease* and Fig. 2 shows the resulting values for all attributes. One can clearly see that the attributes linked to differences between the observed and expected signals are the most important attributes and, in fact, they confirm the selection of attributes used for the traditional PID method, which is based exactly on those metrics. Nevertheless, the results also show that other parameters, *e.g.* TPC signal and covariance matrix components, bear a significant portion of information and can indeed improve the performance of a PID method. Their tuning, however, is cumbersome and therefore not used in the traditional PID, while our automatic classification method based on Random Forests can easily exploit this information to improve PID performance, as we present in the next section.

5.3 Parameter tuning

We tune the hyper-parameters of our approach using a mean cross-validation score of F1 function computed on the binary classification task. Fig. 3 shows the results of the a set of experiments performed with different number of decision trees. One can see that the performance of our Random Forest method for PID saturates for more than 100 trees and we use this value in the following experiments.

Using a similar set of experiments, we tune the other parameters of our method. The final set of values used in the rest of our work can be found below:

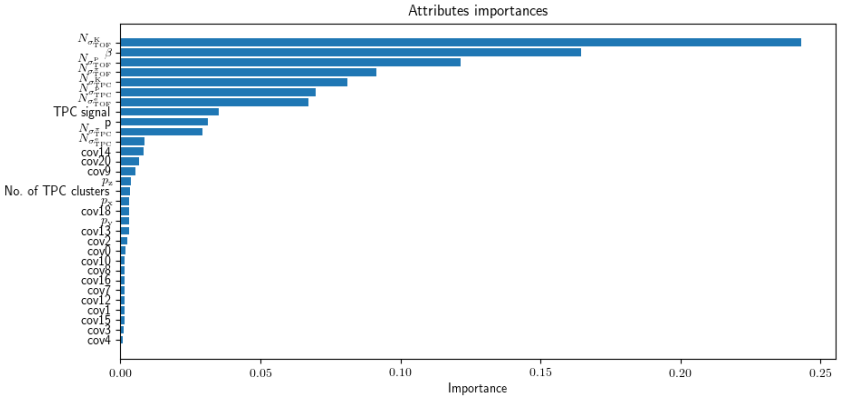

Fig. 2. Contribution of various track parameters to the overall PID from the training of the classifier. The highest importance matches parameters used in traditional PID, which proves correctness of this approach.

- maximum depth of each decision tree: infinite
- minimal impurity of a node for it to be converted into a leaf: 0.0
- maximum number of attributes used per tree training: 5
- minimal impurity decrease of resulting subdatasets for to create a node: 0.0

5.4 Evaluation protocol

As our evaluation metrics, we use standard PID quantities: the *PID efficiency* and *purity*. They are defined in Eq. (1) and Eq. (2), respectively:

$$\text{Efficiency} = \frac{\text{number of tracks correctly classified as a given particle specie}}{\text{number of all tracks of a given particle specie available in the sample}}, \tag{1}$$

$$\text{Purity} = \frac{\text{number of tracks correctly classified as a given particle specie}}{\text{number of tracks classified as a given particle specie}}. \tag{2}$$

Those metrics can be closely related to *precision* and *recall* metrics, widely used in the machine learning community.

5.5 Results

Here, we present the final results of the performance of the traditional PID and our method. Fig. 4 and 5 show a comparison of the efficiency and purity of the traditional and ML-based PID methods of charged kaon selection as a function of

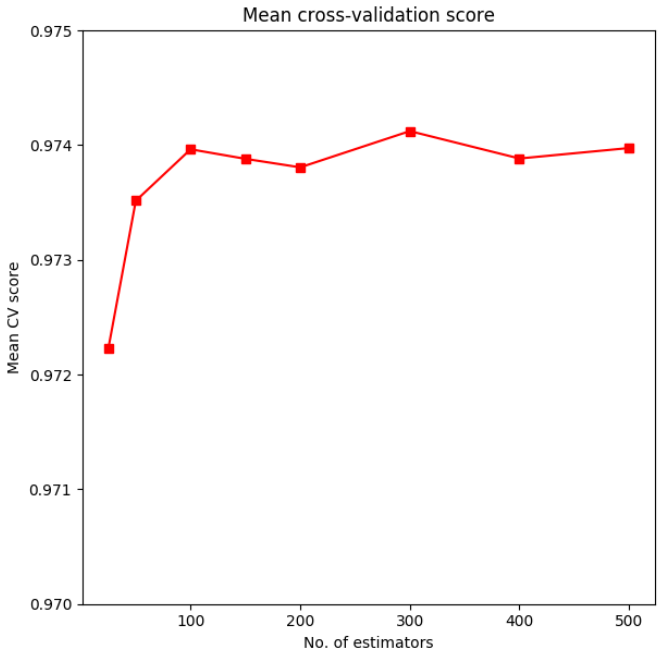

Fig. 3. Mean cross-validation score, using F1-score function to determine the best number of estimators for Random Forest classifier.

particle's transverse momentum p_{T}. Overall, our proposed approach significantly outperforms the traditional method, across both metrics.

The achieved values of PID efficiency and purity, when integrated over p_{T}, are the following:

- for traditional PID the efficiency is 80.68% and the purity of the kaon sample is 88.22%,
- for the Random Forest the obtained efficiency is 97.31% and the purity of the kaon sample is 97.55%.

While maintaining similar or higher purity levels, the Random Forest classifier achieves much higher overall efficiency than the competing traditional PID method. It is especially visible for particles with transverse momentum p_{T} inside range from 0.5 GeV/c to 2.0 GeV/c, where the performance improvement of the Random Forest reaches up to 20% in terms of PID efficiency. Furthermore, the Random Forest achieves over 35% boost in purity over the traditional method for particles with transverse momentum p_{T} higher than 2.0 GeV/c. Overall, the proposed Random Forest method provides statistically relevant improvement

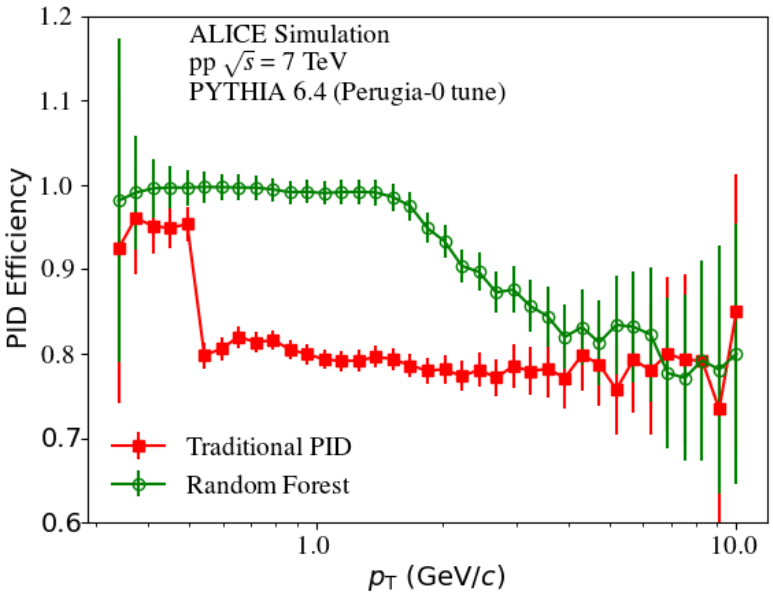

Fig. 4. Comparison of PID efficiency as a function of p_T of the kaon selection between the traditional PID method and the Random Forest classifier.

across both metrics and transverse momentum levels over the traditional PID method.

6 Conclusions

In this paper, we compared the traditional PID procedure with a novel approach that is based on a machine learning classifier. Our evaluation results show that the approach based on a Random Forest classifier achieves higher purity and efficiency levels in the context of binary classification task of kaons. The improvement of our method reaches up to 20% in terms of efficiency and over 35% in terms of purity. Not only did our approach yield better results, but it also significantly decreases the amount of human labor that is necessary to achieve this score. With the ability to scale horizontally with distributed systems, our approach seems to offer a promising alternative to the currently used and computationally expensive methods. This is especially true for analyses which require to massive datasets that cannot fit in a memory of a single machine. Thanks to the proposed machine learning-based method, it possible to exploit larger datasets and obtain higher quality classifiers for the particle identification task. One important limitation of our method is that it depends greatly on how well the Monte Carlo simulation describes the real data experiment. Although there

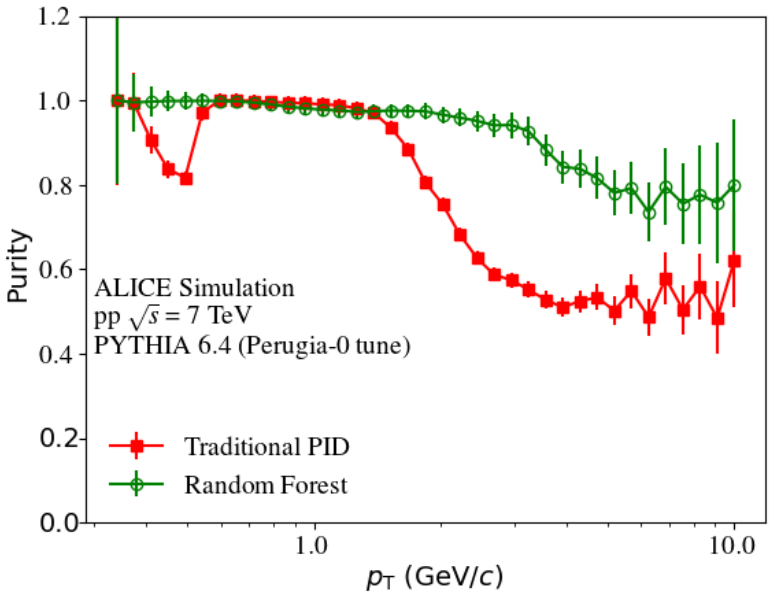

Fig. 5. Comparison of purity as a function of p_T of the kaon selection between the traditional PID method and the Random Forest classifier.

is still place for improvement, we believe that the results presented in this paper can serve as a proof of the potential offered by machine learning algorithms and the Random Forest classifier in particular. Given its performance boost, we are certain that this kind of machine learning approaches can be successfully incorporated in the ALICE Experiment, increasing the quality of high-energy physics results.

Acknowledgements

The authors acknowledge the support from the Polish National Science Centre grant no. UMO-2016/21/D/ST6/01946.

References

1. K. Aamodt *et al.*, "The ALICE experiment at the CERN LHC," *JINST*, vol. 3, p. S08002, 2008.
2. L. Evans and P. Bryant, "LHC Machine," *JINST*, vol. 3, p. S08001, 2008.
3. E. V. Shuryak, "Quark-Gluon Plasma and Hadronic Production of Leptons, Photons and Psions," *Phys. Lett.*, vol. 78B, p. 150, 1978. [Yad. Fiz.28,796(1978)].

4. J. Adams *et al.*, "Experimental and theoretical challenges in the search for the quark gluon plasma: The STAR Collaboration's critical assessment of the evidence from RHIC collisions," *Nucl. Phys.*, vol. A757, pp. 102–183, 2005.
5. L. Breiman, "Random forests," *Machine Learning*, vol. 45, pp. 5–32, Oct 2001.
6. G. Dellacasa *et al.*, "ALICE: Technical Design Report of the Time Projection Chamber," *CERN-OPEN-2000-183, CERN-LHCC-2000-001*, 2000.
7. G. Dellacasa *et al.*, "ALICE technical design report of the time-of-flight system (TOF)," 2000.
8. W. Blum, L. Rolandi, and W. Riegler, *Particle detection with drift chambers*. Particle Acceleration and Detection, 2008.
9. J. Adam *et al.*, "Determination of the event collision time with the ALICE detector at the LHC," *Eur. Phys. J. Plus*, vol. 132, no. 2, p. 99, 2017.
10. J. Adam *et al.*, "Particle identification in ALICE: a Bayesian approach," *Eur. Phys. J. Plus*, vol. 131, no. 5, p. 168, 2016.
11. T. K. Ho, "The random subspace method for constructing decision forests," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, pp. 832–844, Aug 1998.
12. CERN, "Worldwide lhc computing grid." http://wlcg-public.web.cern.ch/.
13. W.-Z. Lin, J.-A. Fang, X. Xiao, and K.-C. Chou, "idna-prot: Identification of dna binding proteins using random forest with grey model," *PLOS ONE*, vol. 6, pp. 1–7, 09 2011.
14. R. Casanova, S. Saldana, E. Y. Chew, R. P. Danis, C. M. Greven, and W. T. Ambrosius, "Application of random forests methods to diabetic retinopathy classification analyses," *PLOS ONE*, vol. 9, pp. 1–8, 06 2014.
15. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
16. W. McKinney, "pandas: a python data analysis library." http://pandas.pydata.org/.
17. T. Sjostrand, S. Mrenna, and P. Z. Skands, "PYTHIA 6.4 Physics and Manual," *JHEP*, vol. 05, p. 026, 2006.
18. P. Z. Skands, "Tuning Monte Carlo Generators: The Perugia Tunes," *Phys. Rev.*, vol. D82, p. 074018, 2010.
19. R. Brun, F. Bruyant, F. Carminati, S. Giani, M. Maire, A. McPherson, G. Patrick, and L. Urban, "GEANT Detector Description and Simulation Tool," 1994.