Use of machine learning methods for particle identification in nuclear physics experiments

Łukasz Sawicki Krzysztof Palmi

March 17, 2021

Physical experiments related to collisions of heavy ions accelerated to ultrarelativistic energies have enormous importance in research of nuclear matter in extreme conditions - high temperature and energy density, close to how it was at the beginning of the Universe. Data collected during just one collision outperforms computational capabilities of many daily used computers. One of the main problems encountered in those experiments is difficulty of differentiating between signal we are interested in and background, produced by every particle, that is not in our interest. This problem mainly applies to heavy, rare and fugacious particles. As one of the solutions comes the use of machine learning allowing for algorithm learning of recognition between particles and background.

`https://github.com/Lukaszz99/ML_HEP`.

1 Generation of rapidity and pseudorapidty

1.1 Objective

Objective of this exercise is to draw histograms of rapidity y and pseudorapidity η of π^0, K^0 i D^0 particles, which mass are shown below.

- $m_{\pi^0} = 134.98 MeV/c^2$
- $m_{K^0} = 497.65 MeV/c^2$
- $m_{D^0} = 1864.84 MeV/c^2$

1.2 Algorithm of performing the task

Following order of execution was implemented:

1. Draw y i p_T n times. y is drawn from range (-5, 5), p_T from range (0, 2000) MeV.
2. Calculate $p_z = m_T \sinh y$, where $m_T^2 = m^2 + p_T^2$
3. Calculate $\cos\theta = \frac{p_z}{p} = \frac{p_z}{\sqrt{p_T^2 + p_z^2}}$
4. Calculate pseudorapidity $\tanh\eta = \cos\theta \rightarrow \eta = \operatorname{arctanh}(\cos\theta)$
5. Make histograms for y and η.

Using the algorithm we obtained the following results.

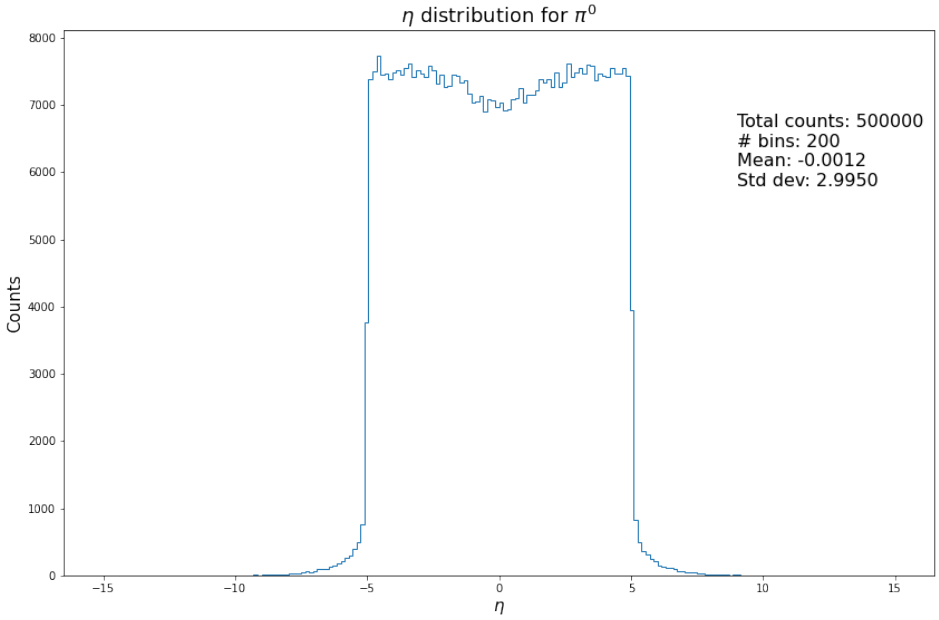


Figure 1: Pseudorapidity η for π^0

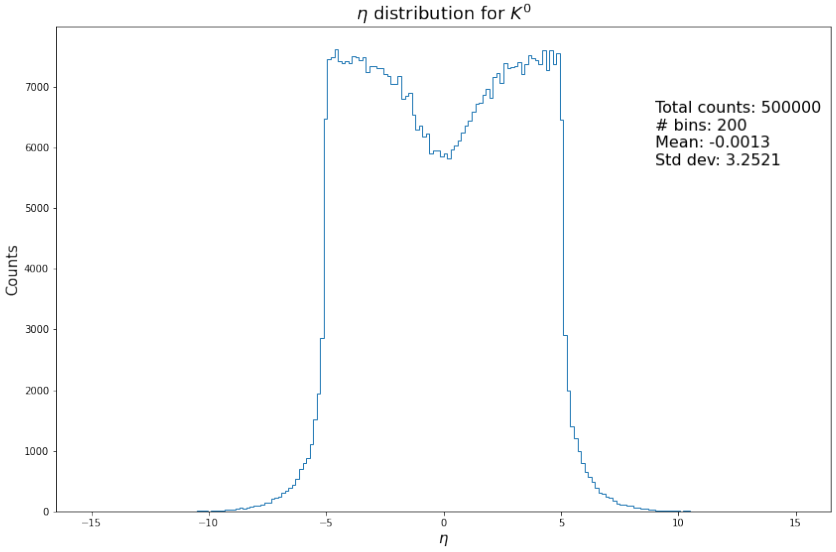


Figure 2: Pseudorapidity η fo K^0

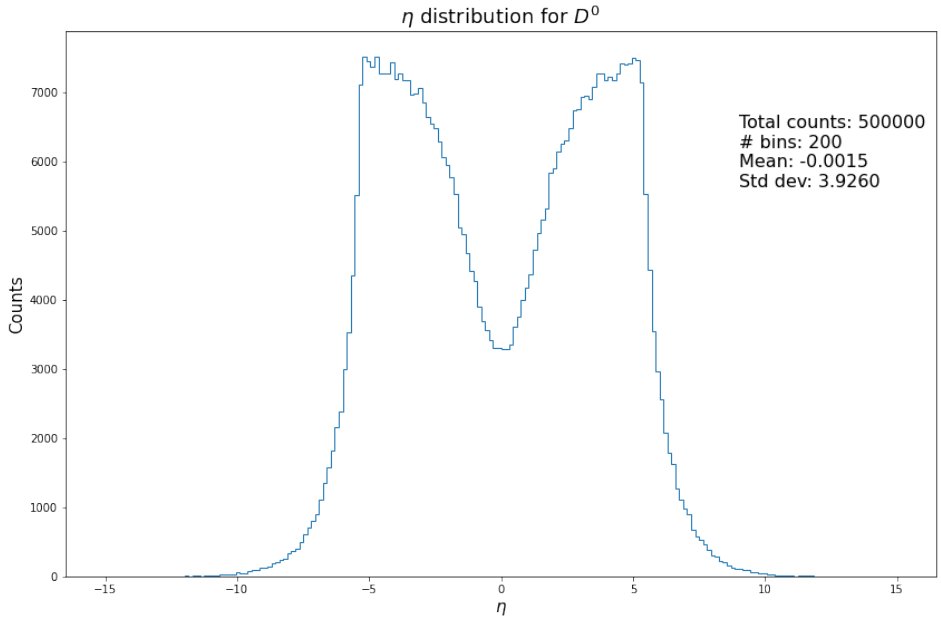


Figure 3: Pseudorapidity η fo D^0

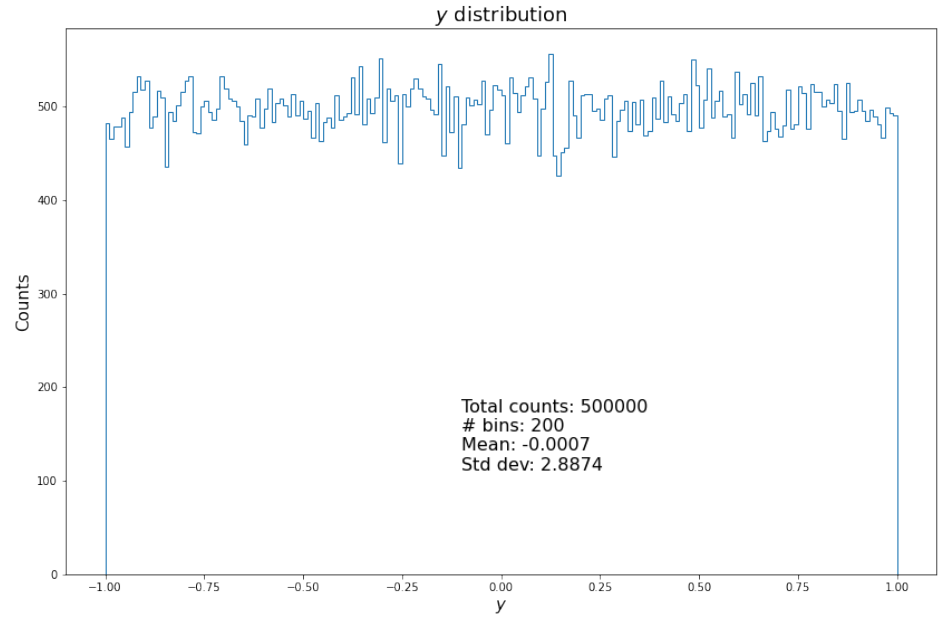


Figure 4: Drawn by lots rapidity y

1.3 Conclusion

Comparing the results of implementing the algorithm we can clearly see, that heavier particles tend to have less "concentrated" pseudorapidity η around mean value $< \eta >$. Moreover, two symmetrical to mean value peaks are created and for heavier particles the distance between them is bigger.

2 Loading data and analysis

2.1 Objective

The first step of this task was to load .root files and display histograms of data (e.g. *pt*, *ptPion*, *ptKaon*, *deacyLength*) located inside files. In the second step we have to analyze correlation between pairs of variables. In order to create effective machine learning models, there is need to select which features should be use in model to balance accuracy and speed of learning model. For this purpose we can analyze features (LDA, PCA, kPCA, Random forest feature importance etc) to decide which are the most important in particle recognizing.

Data used in this section was chosen based on transverse momentum ranging between 1-2 GeV.

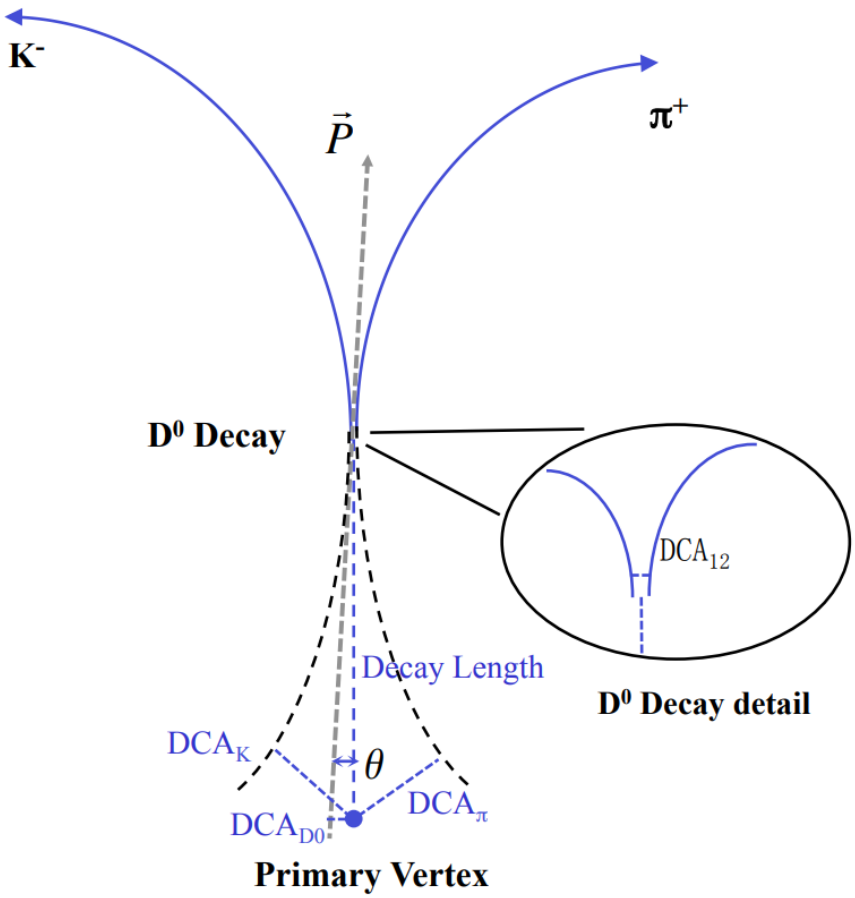


Figure 5: Scheme of $D^0 \rightarrow K^- + \pi^+$ decay and definition of topological variables. Source: https://arxiv.org/pdf/1812.10224.pdf

2.2 Software

To load data, we used PyROOT. This framework allows use of all ROOT classes in Python. While data is loaded we translate .root file to NumPy (Numerical Python library) arrays and then make further analysis.

2.3 Comparison between signal and background

2.3.1 Histograms 1D

Creation of simple histograms with weighted data shows distribution of presented parameters in order to show differences between signal and background.

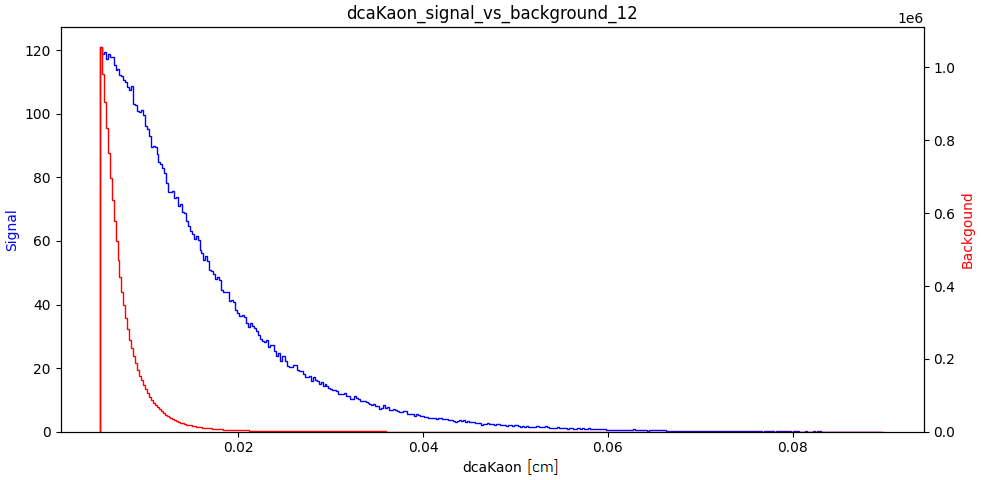


Figure 6: Histogram of dcaKaon.

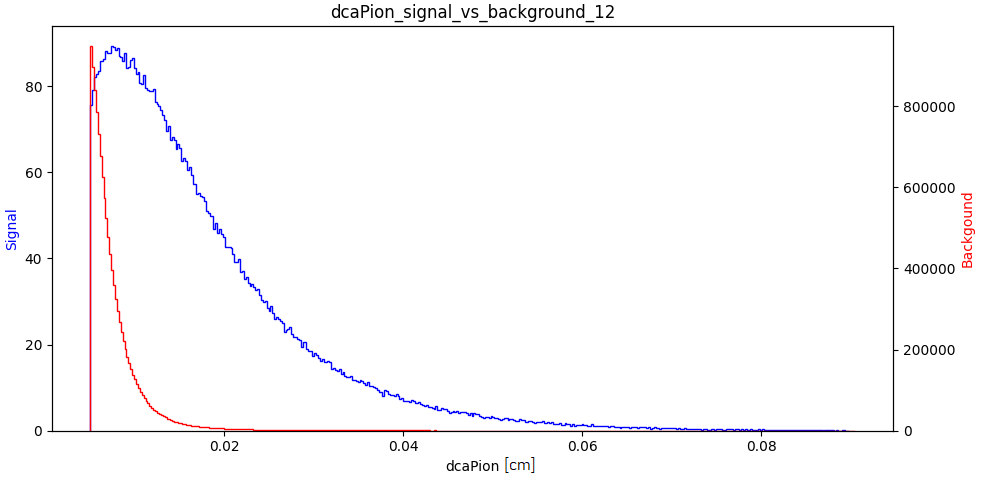


Figure 7: Histogram of dcaPion.

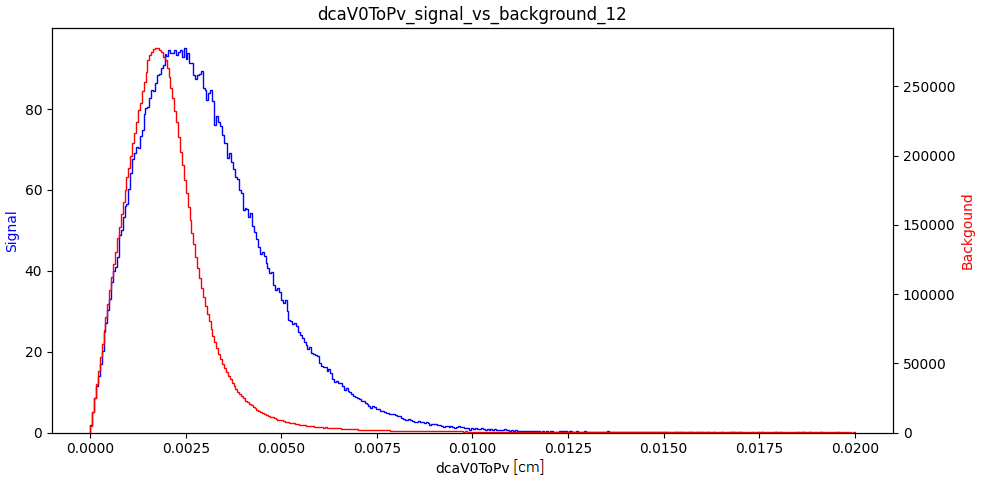


Figure 8: Histogram of dcaV0ToPv.

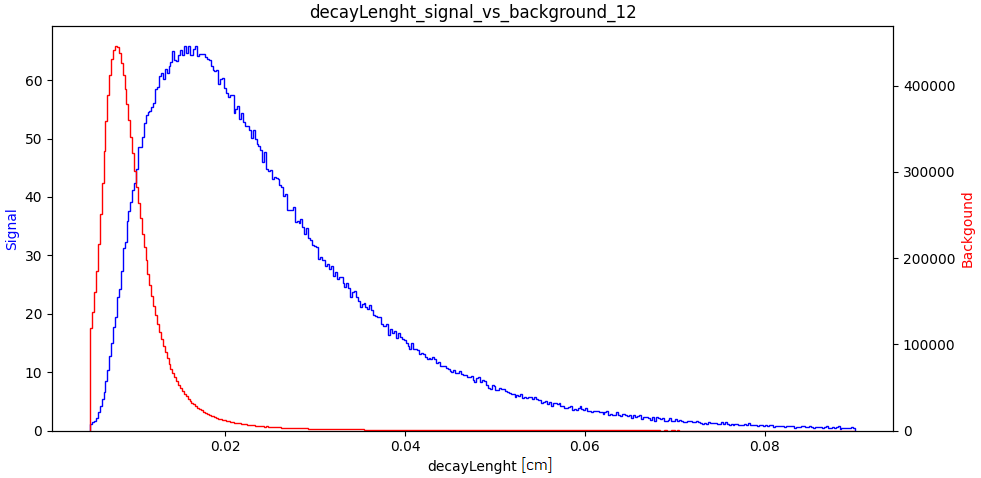


Figure 9: Histogram of decay length.

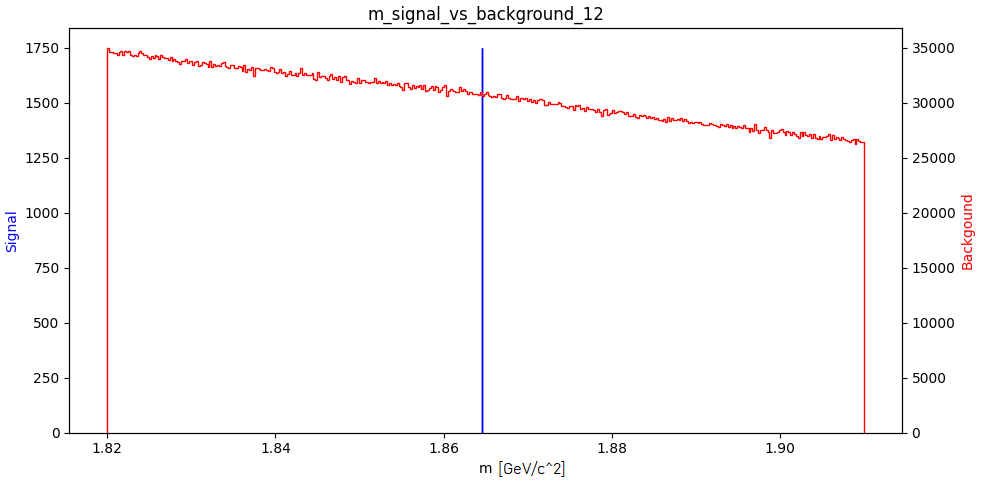


Figure 10: Histogram of mass.

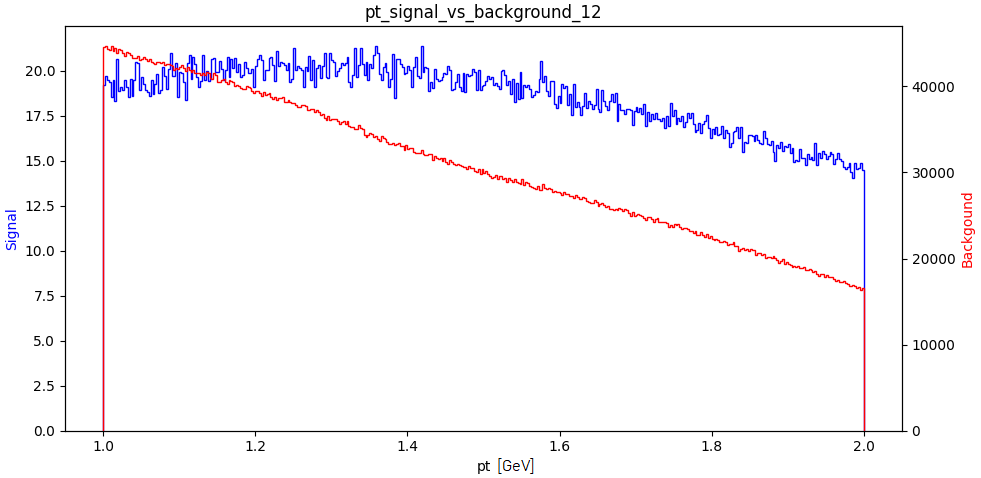


Figure 11: Histogram of dcaKaon.

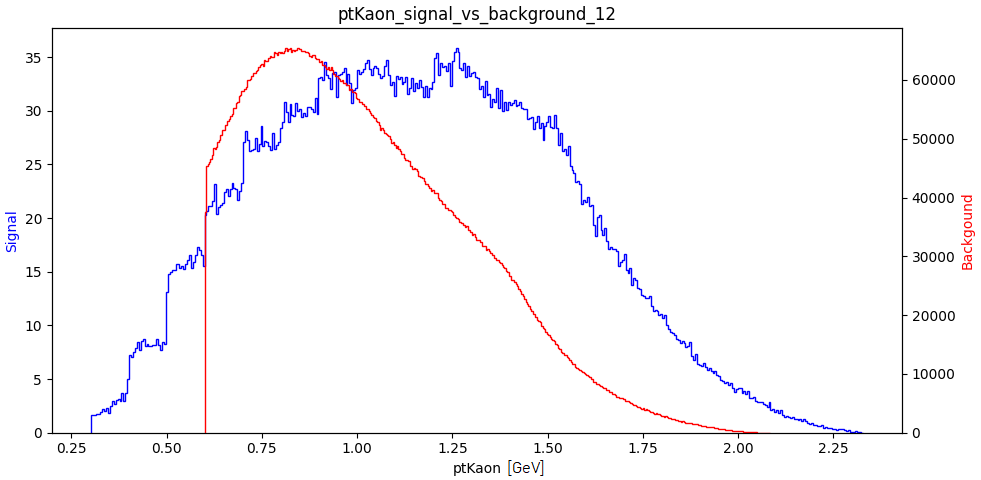


Figure 12: Histogram of dcaKaon.

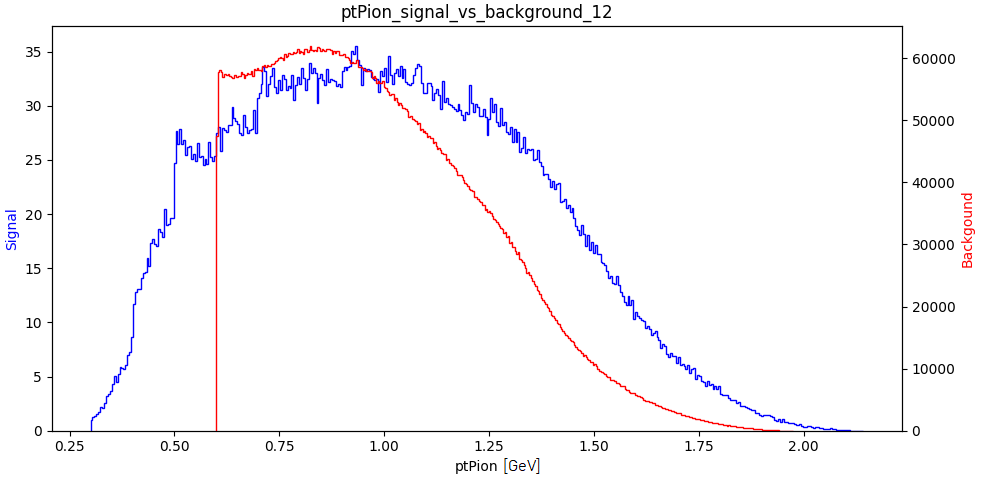
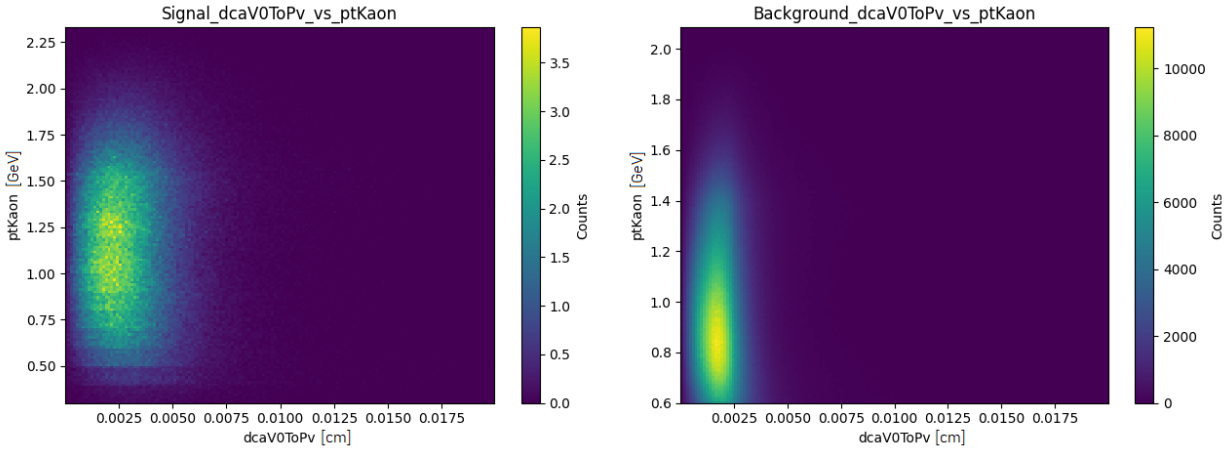
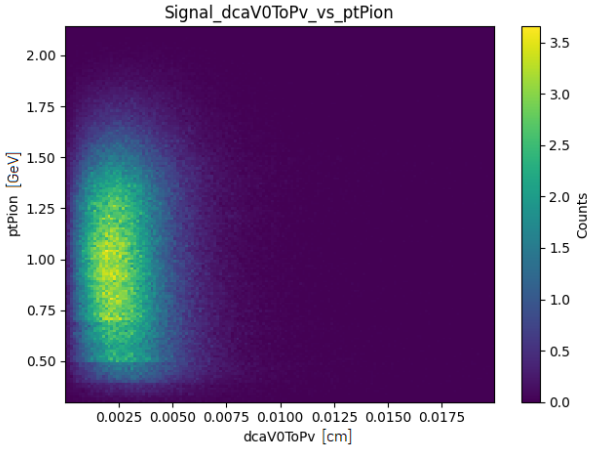


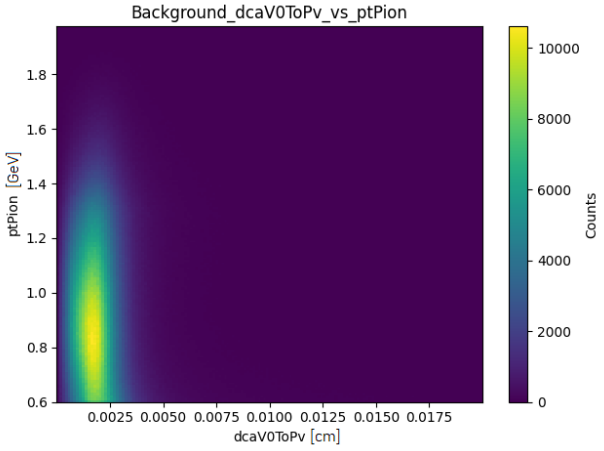
Figure 13: Histogram of dcaKaon.

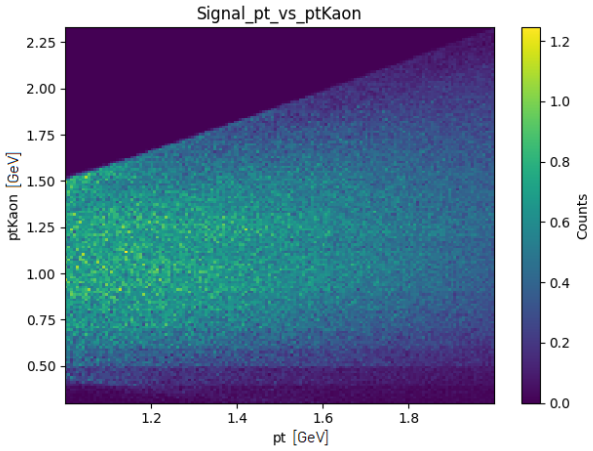
2.3.2 Histograms 2D

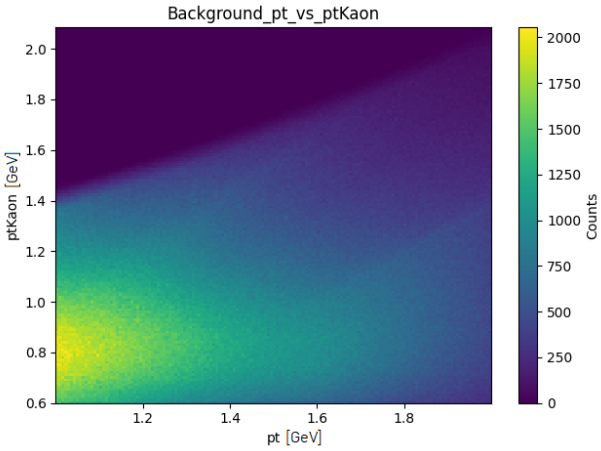
This subsection is mainly created for showing correlations as two dimensional histograms and oppose them to background/signal.

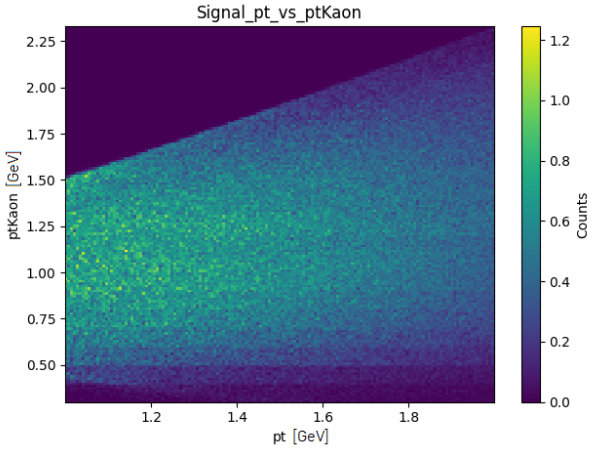


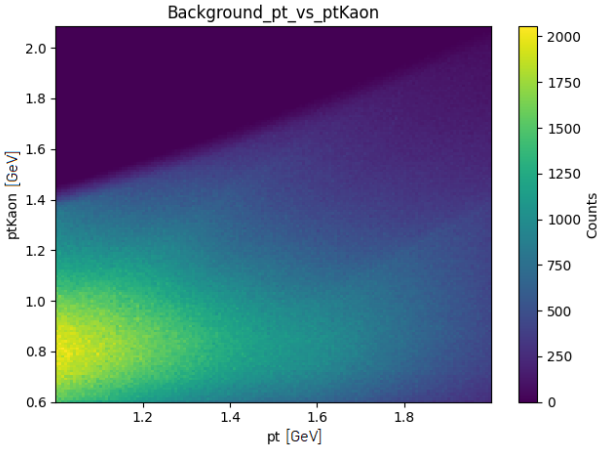


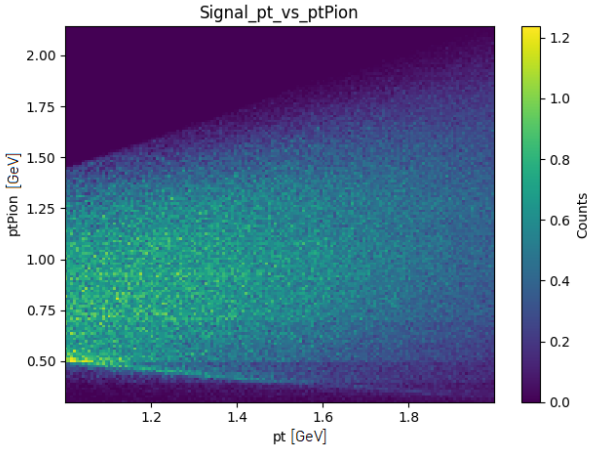


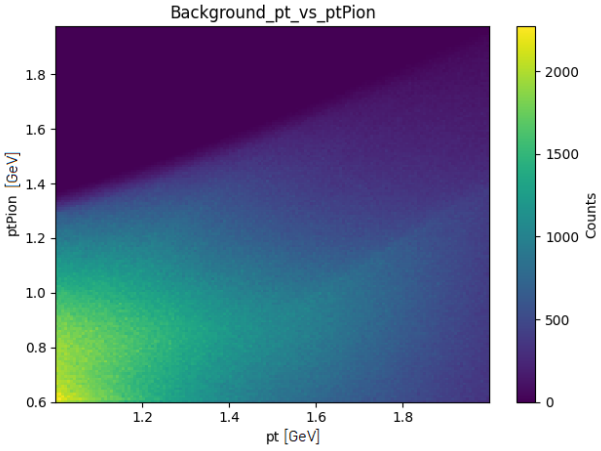

Signal_pt_vs_ptKaon
ptKaon [GeV]
pt [GeV]
Counts

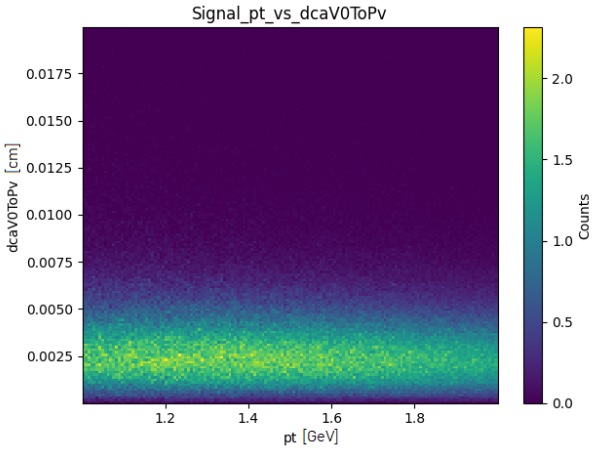

Background_pt_vs_ptKaon
ptKaon [GeV]
pt [GeV]
Counts

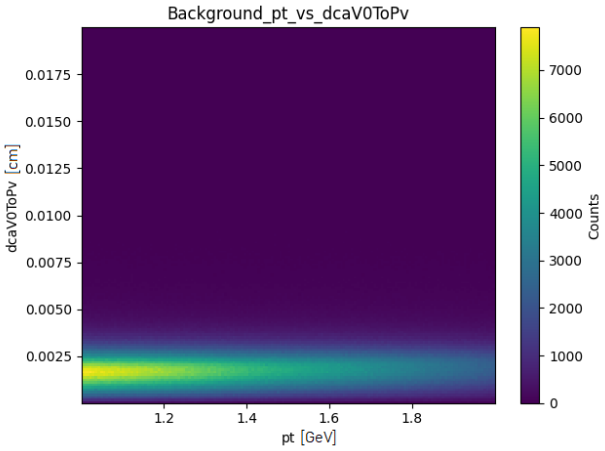

Signal_pt_vs_ptKaon
ptKaon [GeV]
pt [GeV]
Counts

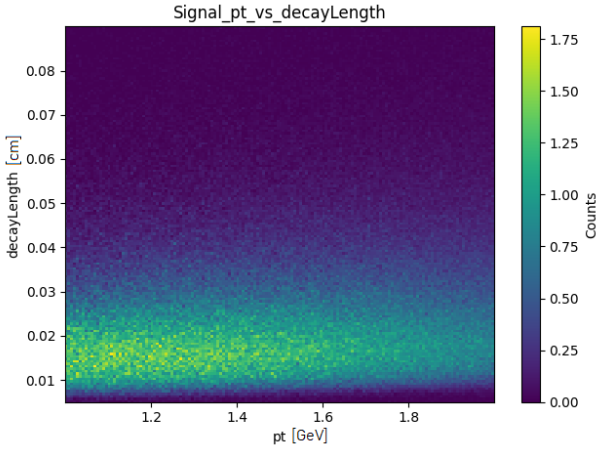

Background_pt_vs_ptKaon
ptKaon [GeV]
pt [GeV]
Counts

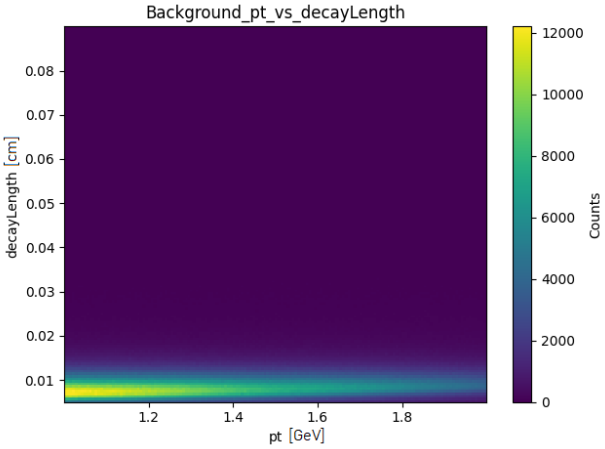

Signal_pt_vs_ptPion
ptPion [GeV]
pt [GeV]
Counts

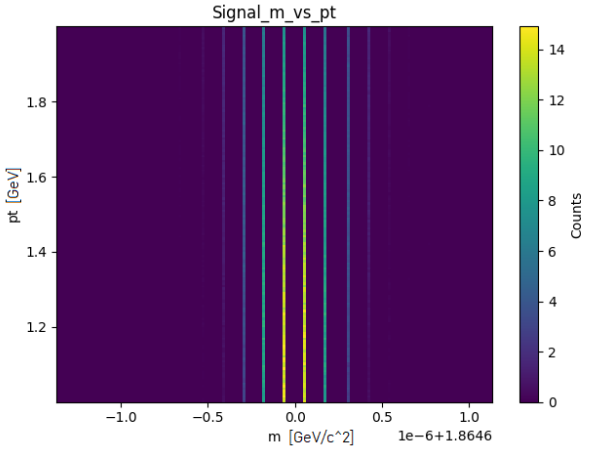

Background_pt_vs_ptPion
ptPion [GeV]
pt [GeV]
Counts

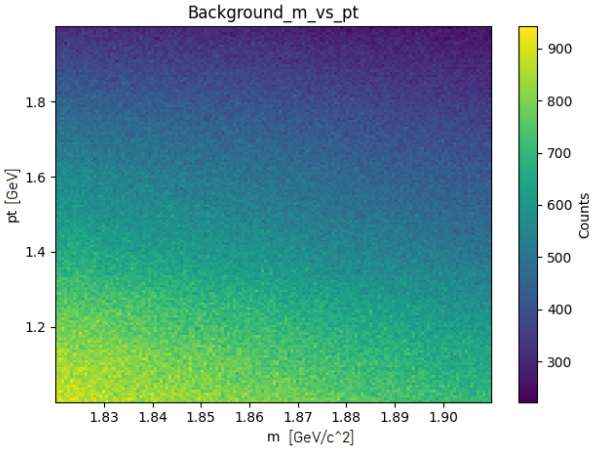

Signal_pt_vs_dcaV0ToPv
dcaV0ToPv [cm]
pt [GeV]
Counts

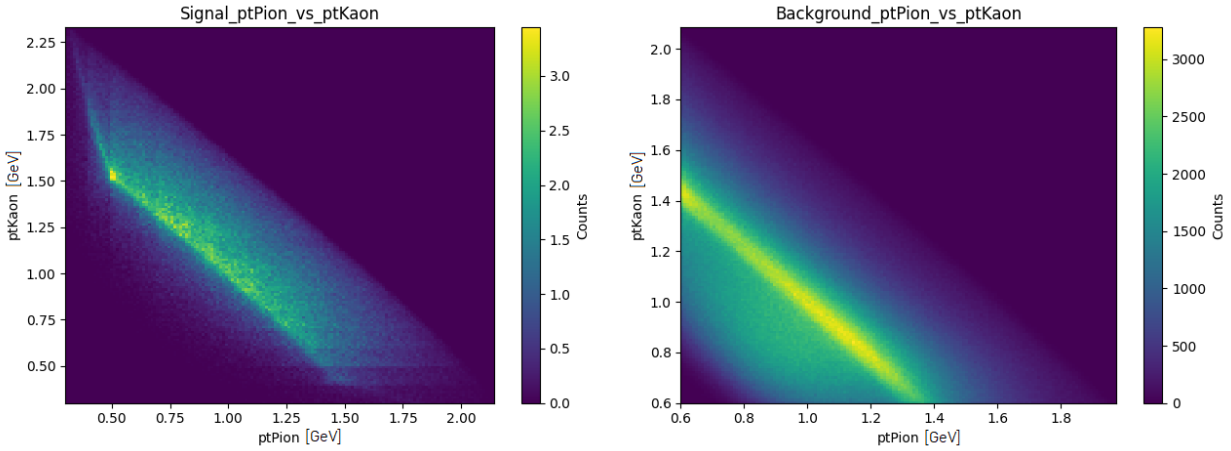

Background_pt_vs_dcaV0ToPv
dcaV0ToPv [cm]
pt [GeV]
Counts


Signal_pt_vs_decayLength
decayLength [cm]
pt [GeV]
Counts


Background_pt_vs_decayLength
decayLength [cm]
pt [GeV]
Counts


Signal_m_vs_pt
pt [GeV]
m [GeV/c^2]
1e−6+1.8646
Counts


Background_m_vs_pt
pt [GeV]
m [GeV/c^2]
Counts



2.4 Correlations in signal data

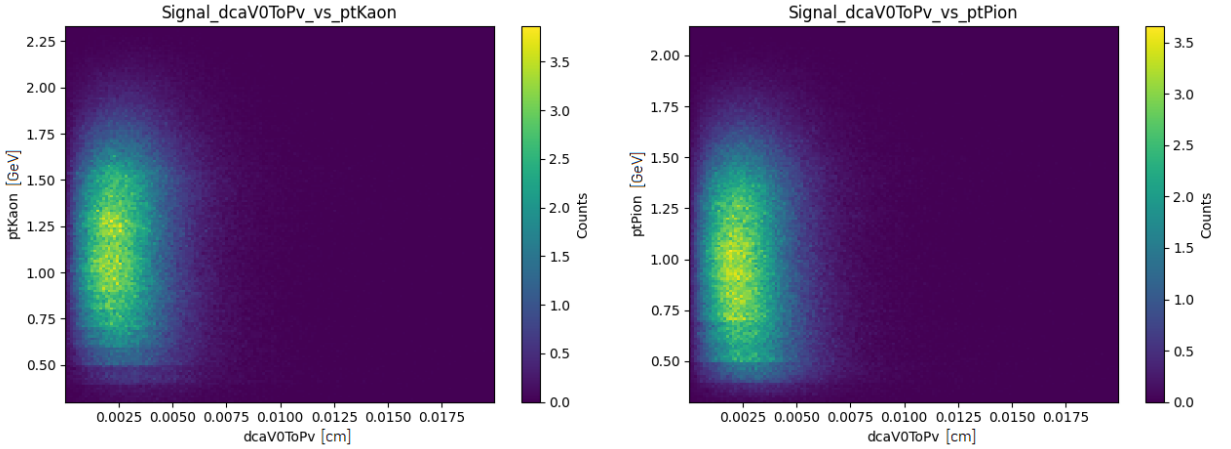


Figure 14: Signal_dcaV0ToPv and transverse momentum of Kaon (left), Pion (right).

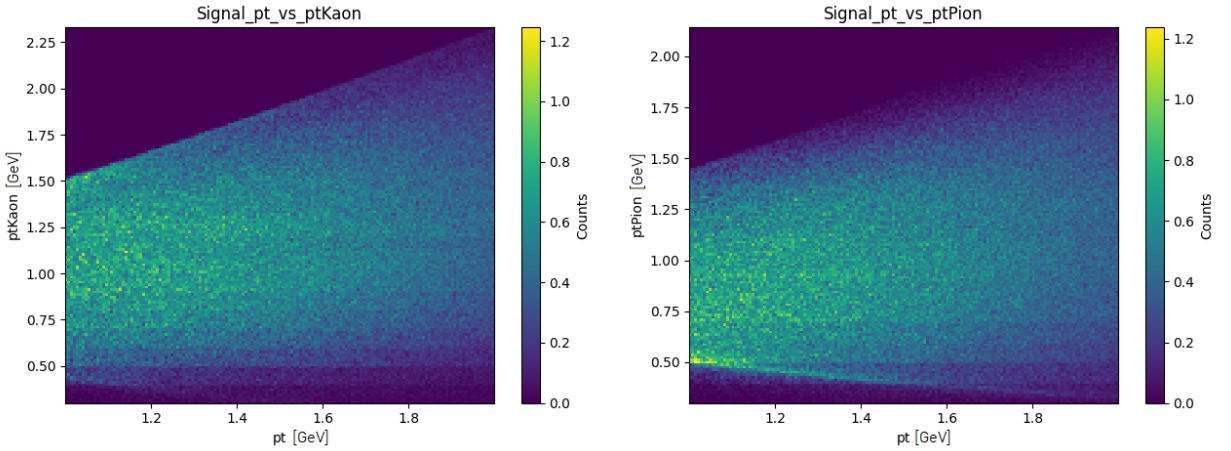


Figure 15: Signal_pt and transverse momentum of Kaon (left), Pion (right).

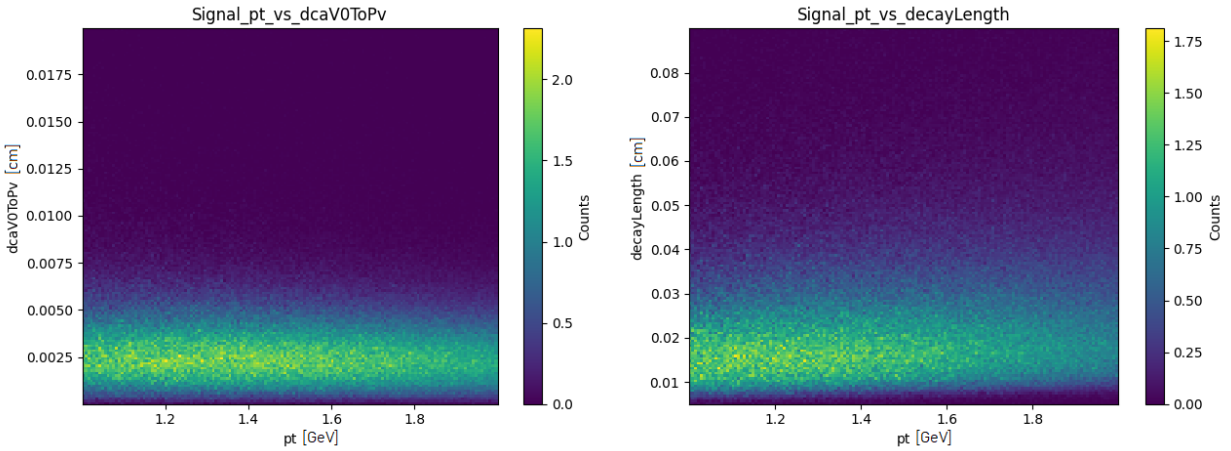


Figure 16: Signal_pt and dcaV0ToPv (left), length of decay (right).

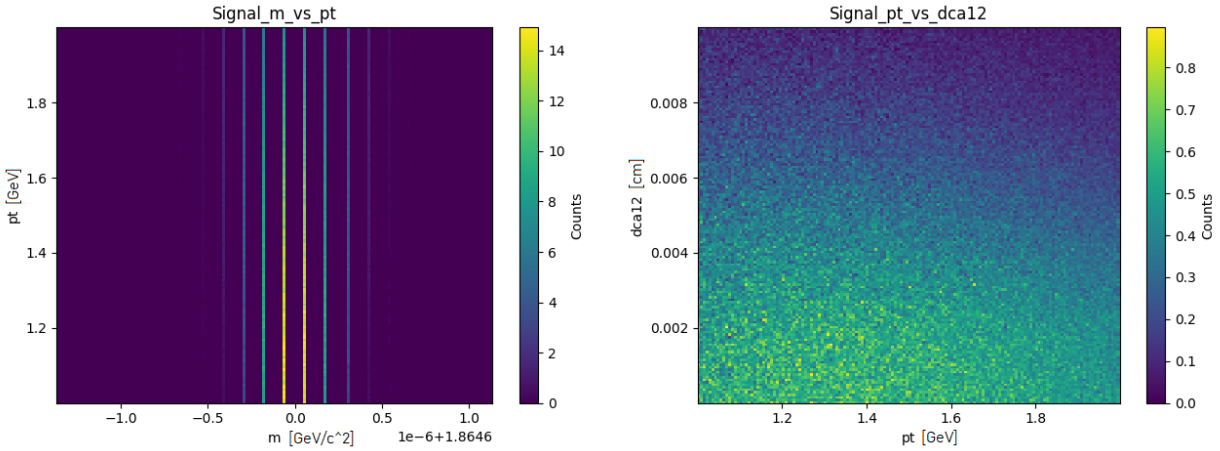


Figure 17: Signal_pt and mass (left), Signal_pt and dca12 (right).

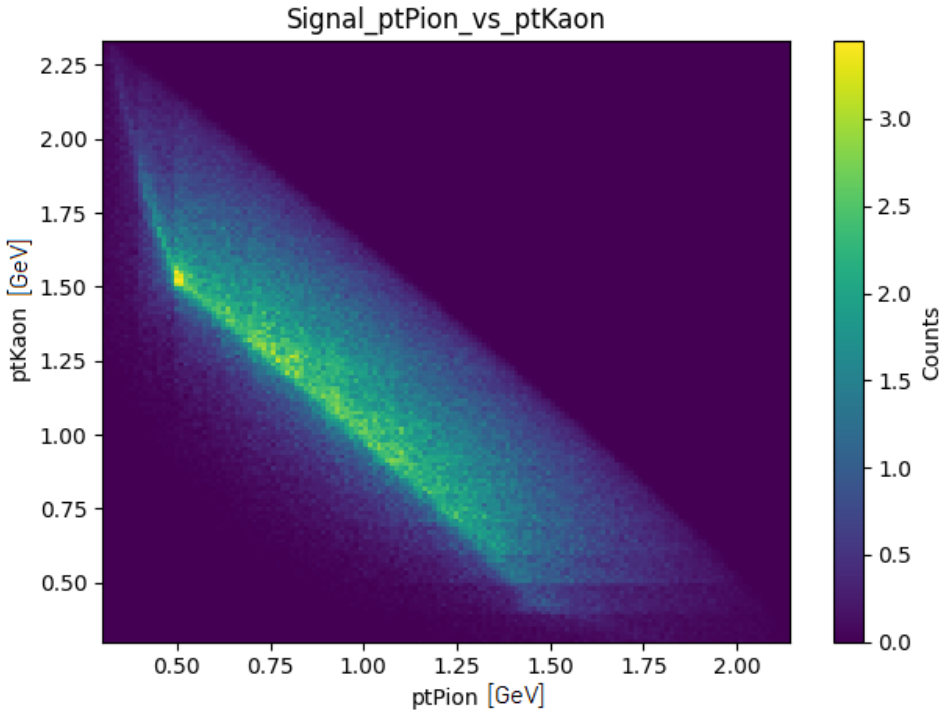


Figure 18: Correlation between transverse momentum of Pion and Kaon.

3 Machine learning models

3.1 Objective

In this section we are trying to use different machine learning models and check their efficiency and predictability. To achieve those results we will be using Python3 package scikit-learn v0.24.0.

3.2 Dataset

Dataset used to training a smaller version of .ROOT file with *pt* range between 1 and 2 GeV. Size of dataset is about 100k rows (particles). We decided to train models on 2 different versions of this set. Difference is inside ratio between signal and background particles. In the first version ratio is 1:2 (signal to background), but in the second- ratio is 1:350. In real-world problem ratio is about 1:500 or even 1:1000, but we have to use 1:350 ratio, because of the dataset size. Every model was trained with 10k learning samples, so it means we used about 30 signal particles (1:350 ratio). It is easy to see, that with ratio 1:1000 there will be only 10 signal particles. It is definitely not enough to train any model.

3.3 Model training

Every model was build with data pipeline. It is structure that handles data manipulation. Our pipeline was very simple. First step was Standard Scaler class, second was PCA analysis. After that, data was transferred to a model. Decision tree model did not have Standard Scaler class. Standard Scaler class transforms every column of data in a way, that average value of column equals zero and standard deviation is equal to one. We trained models with and without PCA on both dataset ratios. Every model was trained with Grid Search method in order to find best hyperparameters. All of the simulations were made on 10k learning samples and 5k of testing samples.

3.4 Model scoring

We were rating trained model with significance score, given by formula

$$S_s = \frac{S}{\sqrt{S+B}}$$

where S is signal ratio - sum of correct classified signal samples divided by number of signal samples in testing set. B is equal to 1 - background ratio. Maximum value of significance score is 1 and it means that every signal, and every background sample was classified correctly, while 0 means the opposite. We decided to change a little our classic formula for significance, to be able to compromise datasets with different signal to background ratio.

3.5 PCA analysis

Principal component analysis (PCA) is the process of computing the principal components and using them to perform a change of basis on the data, which allows to decrease amount of parameters used to describe a set of data. Obtained results show, that three components describe 94% of our training set. During model training we were using PCA with three components.

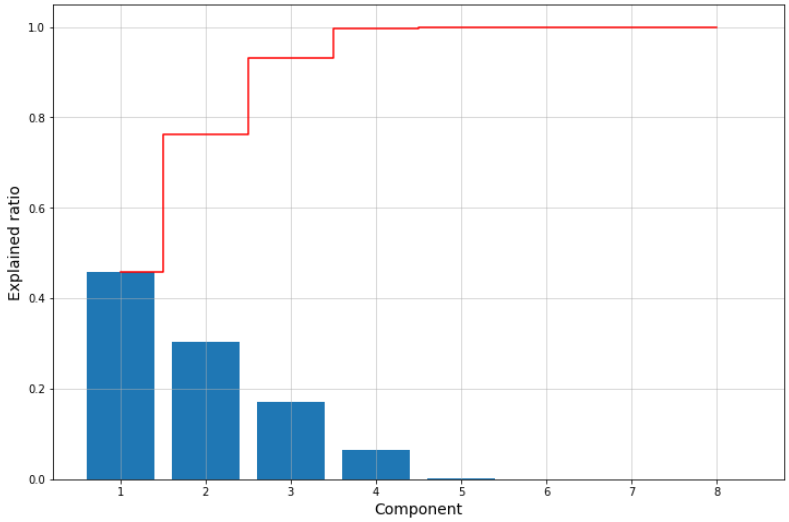


Figure 19: Explanation of a set using PCA.

3.6 Decision Tree

Decision trees are supervised learning method used for classification and regression. They require little data preparation and can be visualised which is a big advantage to understand the process of classification, however they can overfit data when they are left to automatically chose how complex the tree needs to be. Using the grid search method we found that optimal (fast and accurate model) depth of the tree is 10.

3.7 SVC

Support vetor machines (SVMs) are a set of supervised learning methods, which are effective in high dimensional spaces. They are great with non-linear datasets. Their disadvantage is training time due to complex mathematical model. Duirng hyperparameters tuning we found best parameters for every dataset ratio: C = 1000, gamma = 0.01, class_weight = balanced. Last parameter is not related to weight of every sample, but it tells, that dataset has uneven distribution of classes. It increases model accuracy on this type of datasets.

3.8 Logistic Regression

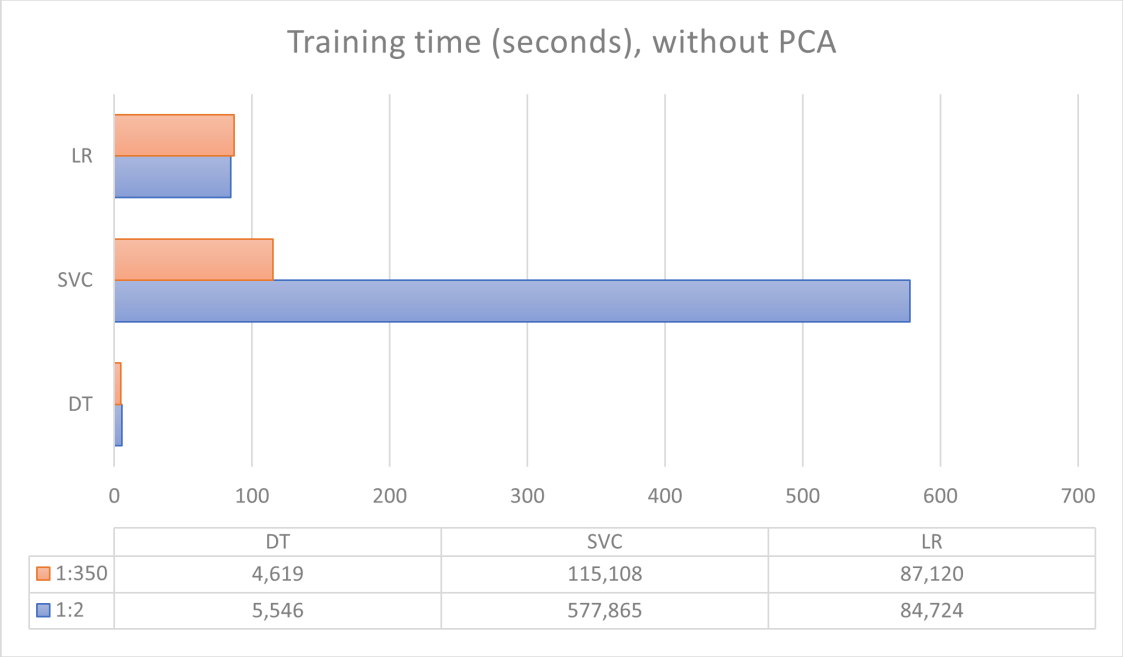
Logistic Regression is a multiclass classifier (the name indicates a regression model, but it is not). Big advantage of this model is training speed, but it is less accurate than more complicated models. During training we found best parameters, presented in table below.

Set ratio	**1:2**		**1:350**	
PCA	True	False	True	False
C	0.1	100	0.01	1000
solver	lbfgs	lbfgs	saga	saga
class_weight	balanced	balanced	balanced	balanced

According to scikit-learn library documentation: **solver** - Algorithm to use in the optimization problem. **lbfgs** - general purpose algorithm. **saga** - faster for large dataset.

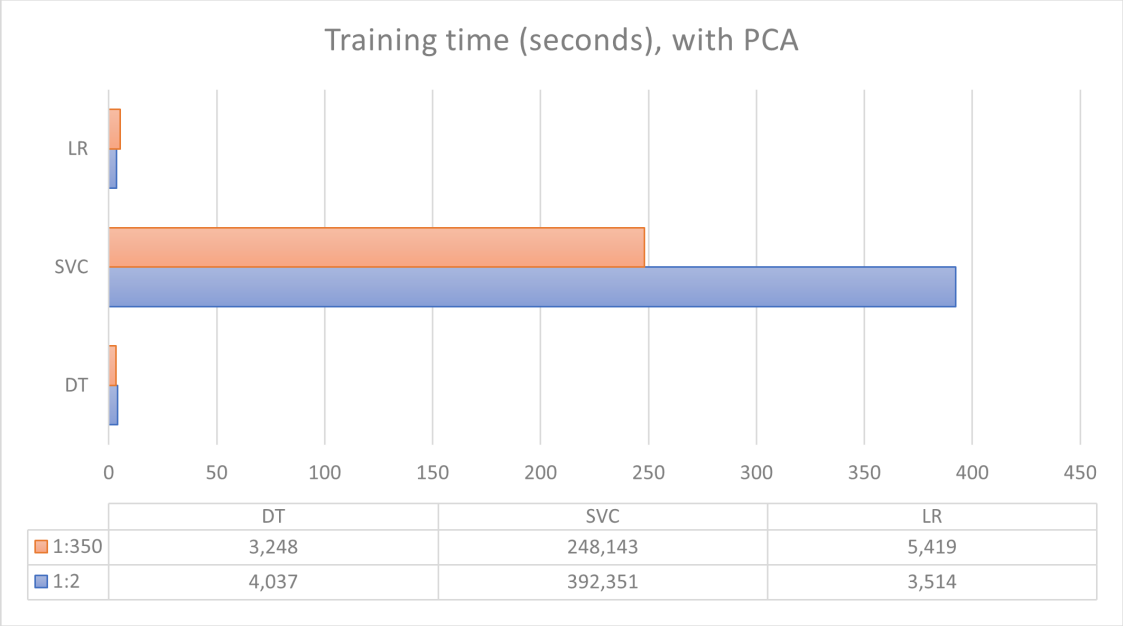
4 Results

4.1 Training time



	DT	SVC	LR
1:350	4,619	115,108	87,120
1:2	5,546	577,865	84,724

Figure 20: Training time dependency based on used model, PCA disabled.

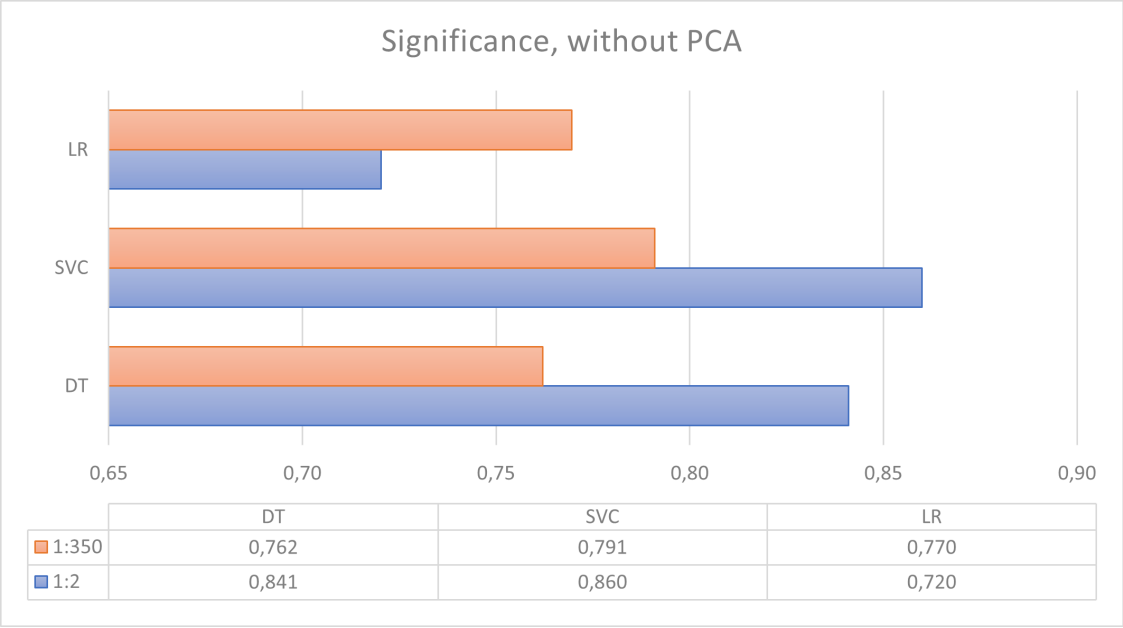


	DT	SVC	LR
1:350	3,248	248,143	5,419
1:2	4,037	392,351	3,514

Figure 21: Training time dependency based on used model, PCA enabled.

Training time gives us information about how much time we need to train our model for it to learn from data we give it. As we can see from both charts the longest period of time needed for training is for SVC model, moreover we can say that decision trees stand out, because how short that model needs for training to be done. PCA (on or off) parameter mostly shortens learning time, but that does not apply to logistic regression.

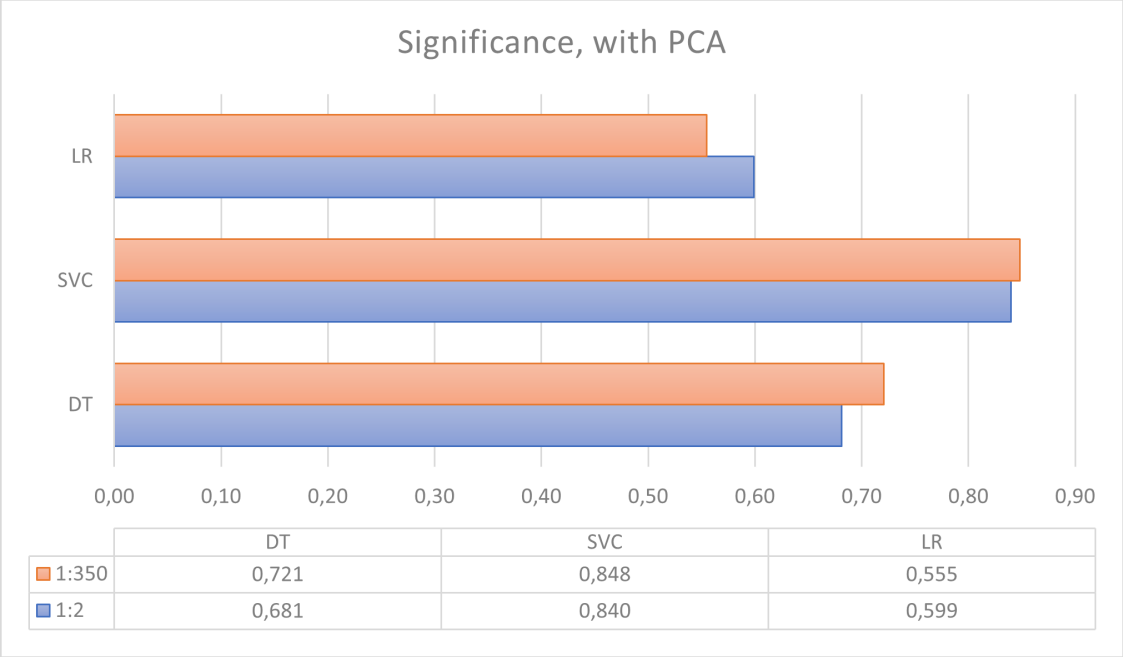
4.2 Model significance score



	DT	SVC	LR
1:350	0,762	0,791	0,770
1:2	0,841	0,860	0,720

Figure 22: Significance score dependency based on used model, PCA disabled.

Despite the SVC model to be the slowest of three, it's significance score is the highest for both PCA options. As a result, use of this classifier must be decided based on importance of significance score vs. training time.



	DT	SVC	LR
1:350	0,721	0,848	0,555
1:2	0,681	0,840	0,599

Figure 23: Significance score dependency based on used model, PCA enabled.

4.3 Conclusion

As we can see from the training time results we can easily say that the most suitable machine learning model for deciding whether data is from background or signal is decision tree. It has the best significance to training time ratio from those three models and is quite easy to control, because of the ease of understanding its parameters through graphical way. If efficiency is much more valuable for us and high training time does not matter it is recommended to use SVC, altough it needs high computing power.

As shown in this project machine learning might be useful for particle identificaion in high energy physics, because of amount of data collected during collisons is too high for person (or group of people) to analyze, but might not be as accurate as specialist, whose efficiency may be close to 100% but time needed for our dataset to be analyzed for that person may be longer than life of average human.