Københavns Universitet
Bachelorstudiet i fysik

Projekt uden for kursusregi 2020

A cool title

Authors:

Jonas Vinther	KU- ID: dlk339
Johann Bock Severin	KU- ID: msk377
Jakob Hallundbæk Schauser	KU- ID: pwn274
Christian Kragh Jespersen	KU- ID: htd809

Advisor:

Troels Christian Petersen	Email: petersen@nbi.ku.dk

This report consist of **30** pages of main text and **???** pages of appendices.

The report has been handed in The 30th of October, 2020.

Abstract

Contents

1 Introduction

The purpose of the project is to investigate the use of Machine Learning algorithm in the field of High Energy Physics. Mainly we will be considering how to treat data from CERN containing different features from collisions, like calculated mass, momentum, angle, position, etc.. This dataset comes in two batches: one collected from the CERN-experiment (data) and one made in Monte Carlo simulations (Simulation).

One question we try to answer is how well you can train a Machine Learning model in data when there is an absence of a truth label. Different methods for this is training models on pseudolabels consisting of cuts in mass, and labels from a trained model in Monte Carlo.

Another big question, is how you determine the quality/accuracy of a model. In this project we tryed to develop a method for drawing ROC-curves and estimating AUC-scores by fitting a peak in data while varying the cutoff criteria from the trained model.

2 Data and theory

2.1 Data and Monte Carlo Simulation

- What do we have? - (?) Atlas detector observations and MC - Fit peaks in data to find mean, variance and similar attributes.

- Describtion of dataset (POSSIBLY NEEDS CORRECTION): Two beams of protons are clashed in the center of the ATLAS detector, the resulting cascades of particles creates hits throughout the detector. From these hits, tracks are generated. For many tracks there exists a common vertex from which these two tracks (particles) were generated (the vertex being a 'decay' of 1 particle to others (note particles includes fotons)). From these tracks and vertices a ((?) perhaps the most likely) k_s^0 (maybe more correctly v0)) candidate is found. This candidate has a primary vertex from which the v0 (?) is created and another vertex at which it decays. Is it thus from this candidate that all the features are calculated. The whole procedure is then repeated again every $50ns$, such that each 'row' in the dataset is one such event and all the events are completely uncorrelated.

2.2 Algorithms of choice

Supervised machine learning is an umbrella term for algorithms that improve automatically by looking at labeled training data. As the main purpose of our project is optimizing classification, the obvious algorithm of choice would be a decision tree.

2.2.1 Boosted decision trees

The way decision trees categorize data is very intuitive. A tree consists of multiple connected branches, each branch 'naively' separating the input according to a specific number of features. When the bottom of the tree is reached, a prediction for the inputs category is returned. When multiple smaller decision trees are trained and combined, the resulting forest is called boosted (ie. a set of weak-classifiers are turned into one strong classifier).
An example of classification of data can be seen in figure 1.

XGBoost There are multiple ways of 'training' the trees ie. optimizing the tree for your specific purpose. One of the most reliable and efficient is found in the XGBoost-library (eXtreme Gradient Boosting). As the name suggests, XGBoost implements gradient boosting which uses regression to quickly produce an ensemble of weaker prediction models.

$$\begin{aligned}
\hat{y}_i^{(0)} &= 0 \\
\hat{y}_i^{(1)} = f_1(x_i) &= \hat{y}_i^{(0)} + f_1(x_i) \\
\hat{y}_i^{(2)} = f_1(x_i) + f_2(x_i) &= \hat{y}_i^{(1)} + f_2(x_i) \\
amp;\dots \\
\hat{y}_i^{(t)} amp;= \sum_{k=1}^{t} f_k(x_i) &= \hat{y}_i^{(t-1)} + f_t(x_i)
\end{aligned}$$

LightGBM Another very popular algorithm is the Microsoft-created Light Gradient Boosting Machine, better known as LightGBM. The main difference between LightGBM and its competitors (XGBoost for example), is the fact that LightGBM uses their a technique called Gradient-based One-Side Sampling (GOSS) finding the optimal cutting value while XGBoost uses a histogram-based algorithm which is slower in some cases. //Apparently wrong// Another very popular algorithm is the Microsoft-created Light Gradient Boosting Machine, better known as LightGBM. The main difference between LightGBM and its competitors (XGBoost for example), is the fact that LightGBM grows trees leaf-wise making it capable of a lower loss with fewer leafs. The difference can be seen in figure **??**
hallo We have used both algorithms but have primarily used the XGBoost implementation from YYYY.

- UMAP (should we have/do we have a umap section?)

- XGBoost / LightGBM (boosted trees)

2.2.2 Shaples Additive Explanation (SHAP) values

Normally, when we train a simple model (decision tree, linear regression, etc.) the model itself explains the result. In a linear regression, we can interpret the results as slope and intercept

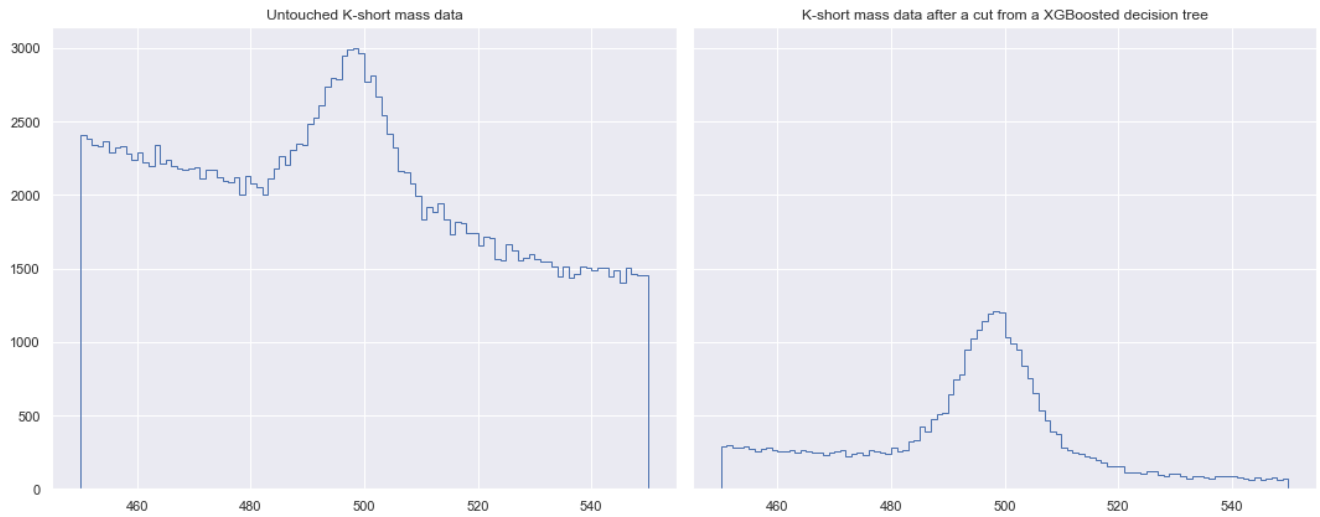

Figure 1: A visualisation of a boosted decision tree labeling data. Left is data around the expected peak directly from CERN. Right is the same data after a ML-cut.

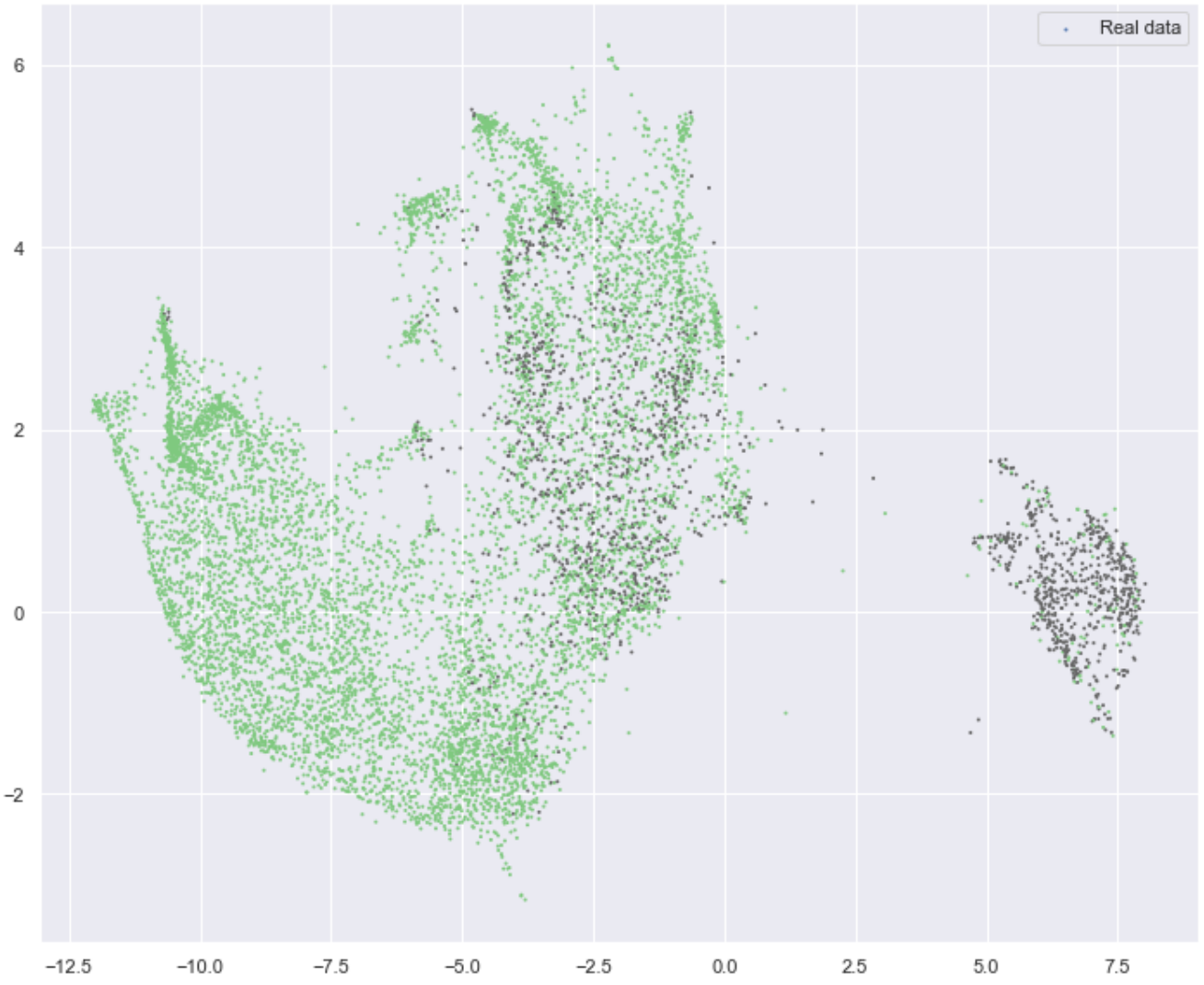

Figure 2: UMAP transforming actual data after supervised training on MC. Dark data points are 'true'-labels from XGBoost.

or in decision trees the different cuts can be written explicitly. However, when we have more complex models like Neural Networks or boosted decision trees, the results are way harder to interpret as the model is very complex. To enlighten us more, an approach is to quantify the importance of the parameters in the model. To do this we are using the SHAP (SHapley Additive exPlanations) algorithm. [1]

The SHAP values are build on the game theoretical idea of shapley values. Which is way to determine the reward an actor should receive according to their contribution. It is calculated for each feature i by training a model with f using the feature space S without i and $S \cup \{i\}$ with the feature i. The Shapley value is now calculated by:

$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} \left(f_{S \cup \{i\}}(x_{S \cup \{i\}}) - f_S(x_S) \right) \tag{1}$$

- Shap

Shap values (an acronym for SHapley Additive exPlanations) is an incredibly useful tool when working with algorithmic predictors like decision trees. The Shap values show the impact each feature in the input space had on the final prediction, making the machine learning much less of a black box and providing us with valuable information on the subtleties of the data. Shap works by running a model multiple times while changing the inputs in the parameter space. This allows it to reach a numerical estimate of the relative importance of any given input.

A simple example of the results of a Shap-analysis can be seen in figure **??**.

2.2.3 Correlations and Maximal Information Coefficient

In this project, we have multiple uses of correlations: to select parameters that contain all the information; minimize correlation with mass, as we use it to define labels for our boosted trees and to generate decorrelated feature spaces, so we can compare results of models with each other. A normal approach would be to use the Pearson correlation coefficient defined by:

$$\rho_{x,y} = \frac{\mathrm{cov}(x, y)}{\sigma_x \sigma_y} \tag{2}$$

This method is computationally efficient and is good at finding linear correlations. Thus it would have been a nice approach had we used a linear classifier such as LDA.

However, the primary method used for classifying is boosted decision trees (LightGBM and XGBoost), which are far from linear models. and are thus sensitive to non-linear correlations. To calculate more general correlations we use the Maximal Information Coefficient (MIC) [2] in this project. This correlation builds on the mutual information, which is defined by:

$$I(X, Y) = \int dxdy \; p(x, y) \log_2 \frac{p(x, y)}{p(x)p(y)} \tag{3}$$

Which compares probability distribution over two variables. A more illuminating form is to write out the logarithm to: $\log_2 p(x,y) - \log_2 p(x)p(y)$ which can be seen as the bitewise difference in the information contained in the joint probability between x and y compared to the assumption that x and y are independent variables $p(x)p(y)$ [1]. Computationally the mutual information $I(X,Y)$ is estimated by binning the data. Binning over X, Y bins gives:

$$I(X,Y) \approx \sum_{X,Y} p(x,y) \log_2 \frac{p(x,y)}{p(x)p(y)} \tag{4}$$

However, this measure is quite sensitive to the choice of binning, but it's also convex. To make up for ..
– or –
This method would be sufficient if we had neat parameter spaces like normalized gaussian distributions. But as the parameters are very different in probability densities we would experience a big effect from the choice of binning method. To make up for this we use the MIC-algorithm. Which maximizes:

$$\mathrm{MIC}(X,Y) = \max_{|X||Y|<B} \frac{I(X,Y)}{\min(|X|,|Y|)} \tag{5}$$

(Johann): ret notation til The mutual information is thus maximized for different binning schemes that satisfy having sizes less than some number B. The use of MIC to determine correlation now gives us a general correlation metric that could be used for very differently shaped probability distributions. This is however heavy computationally, and we sacrifice the amount of data used to determine the correlation (to ≈ 5000 hits).

- Maximum information coefficient While the sheer number of data points and associated parameters could make any statisticians mouth water, the fact that the endgame is a fit in the mass-parameter gives rise to some interesting problems. Any correlation between a feature used for cutting in and the 'ks_mass'-parameter could muddle(?) the results by either artificially magnifying the peak or unknowingly suppressing it. The goal for any cut is to improve the signal-to-noise-ratio but an operation on a heavily correlated parameter will most likely change the composition of the data. The black-box nature of machine learning algorithms makes it particularly important to be conscious of the feature selection.
The most common measure for correlation is the Pearson Correlation Coefficient which quantifies the linear dependency between two parameters, but breaks down at more complex correlations. We have therefore chosen to primarily use the MIC (Maximal Information Coefficient) even though the implementation is considerably slower

2.3 Methods for model evaluation

2.3.1 ROC-curves without labels

- What is ROC Curve; The ROC curve shows the relationship between the false positive rate and the true positive rate, ie. given some threshold what is the ratio of signal you truthfully

[1] $p(x,y) = p(x)p(y)$ when and only when x and y are independent variables

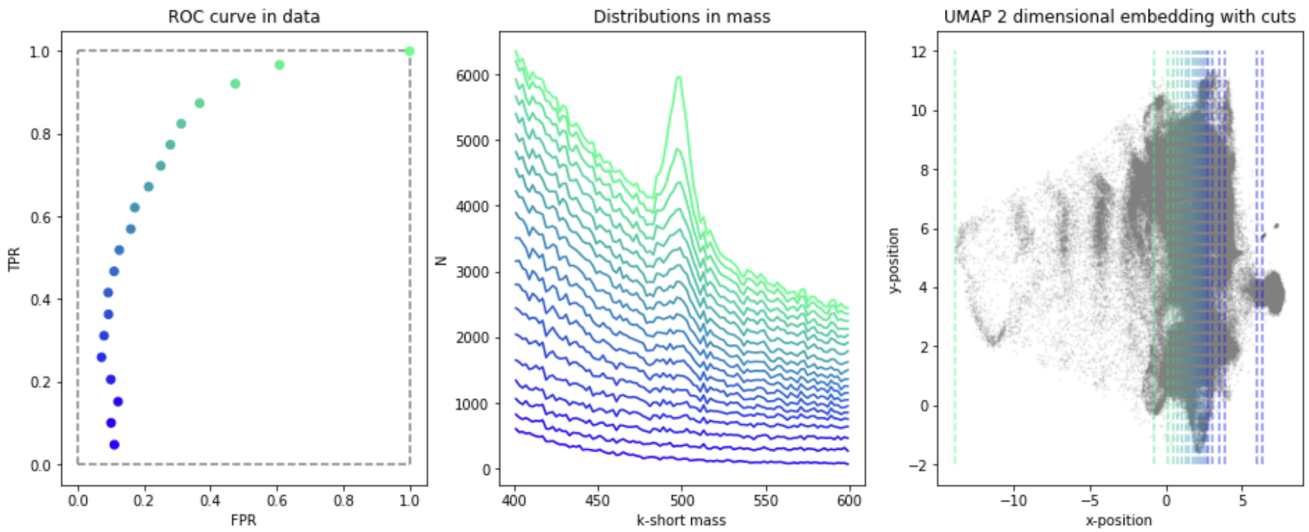

Figure 3: Using supervised UMAP as a classifier based on embedding of data. UMAP is trained on MC and applied to data

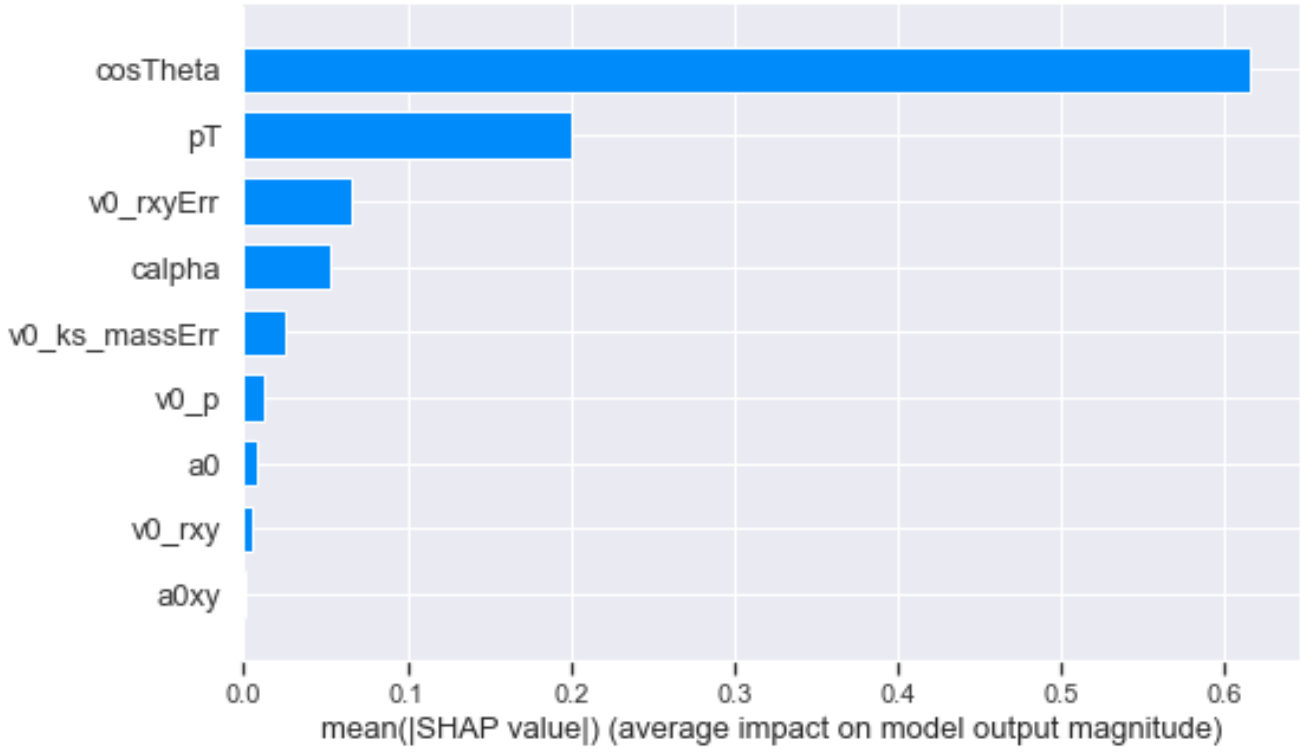

Figure 4: The nine most important parameters according to a SHAP-analysis on one of our simple boosted decision trees

classified as signal compared to how much of the background that is falsely classified as signal. Therefor, given some prediction score, if the ROC curve is straight from beginning to end, the prediction is random, but if the ROC curve is curved towards one of the corners, the prediction scores are useful.
- How to do this without labels; Usually the false/true positive rate is calculated by the use of truth-labels. However, whenever it's possible to estimate the false/true positive rate, it's possible to approximate the ROC curve. In this data, it is possible, through the mass histogram, to estimate the ratio of signal and the ratio of background retained, given some threshold, since the signal shows as some localized peak on the background. The procedure is firstly to find the amount of signal and background found in the test sample by fitting the mass histogram. Thereafter, cuts in the prediction scores are made and once again the mass histogram is fitted to find out how much signal and background is retained. This procedure works best when the mass histogram does not fluctuate much for different threshholds, e.g. if the background is removed 'uniformly', especially around the signal peak. For example, if, at some threshold, the background just outside the peak is removed but the background under the peak is kept, you will erroneously achieve a signal ratio larger than 1.

2.3.2 Correlations with mass

Using the MIC method we compute the score MIC-score between each feature and the mass.

Most correlated with mass are:

Table 1: Table displaying correlation with mass both using the MIC method as well as the linear correlation using the Pearson coefficient

Feature	MIC (Data)	Pearson	MIC (MC)	Pearson (MC)
pT	0.47	0.68	0.47	0.67
v0_ks_massErr	0.38	0.49	0.34	0.57
alpha/Alpha	0.28	-0.02	0.26	-0.07
calpha	0.22	-0.38	0.21	-0.38
v0_rxyErr	0.17	-0.20	0.18	0.19
cosTheta	0.15	-0.03	0.18	-0.05
pL1	0.13	0.13	0.12	0.05
v0_rxy	0.13	-0.14	0.12	-0.10
v0_thetastar	0.13	0.01	0.13	0.08

These features give some issues when used to classify. Since there is not truth label in the Data we use mass to make pseudolabels. However, when the parameters used are correlated with mass, our algorithm of choice can instead classify based on mass. An example of this is shown where the 10 values above are used to illustrate a correlated classifier.

Training a classifier (here lightGBM) on a set of features given in the table above. We obtain a cliassifcation of a test sample found in

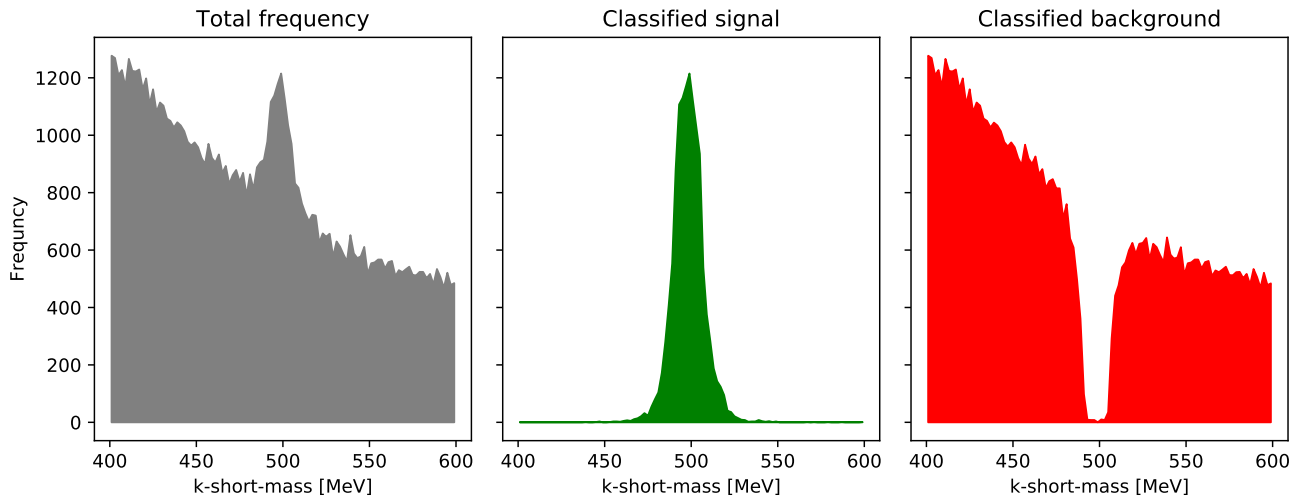

Figure 5: Test sample in Data with mass between 400-600 MeV. The data is seperated using a lightGBM trained with correlated features displayed in the Table 1

Decorrelation? / Effect on ROC-curves without labels

2.4 Parameter selection

2.4.1 ATLAS Detector layout and a priori expectations

- Lay-out of the ATLAS detector + expectation for what parameters from physical reasoning.

"CosTheta" is cosine of the angle between the vectors of the two charged particles which has decayed from a v0 particle that have travelled some distance from the primary vertex. Thus, if particle creation took place at the common vertex, we expect the two tracks to be parallel (or anti-parallel (?) (trueKs have value 1)) due to conservation of momentum.

"v0_rxy" is the distance of the v0 vertex to the primary vertex. Therefor this variable only has a limited range within which it retains physical meaning. The range is determined by the layout of the detector. ($R_{detector}$ 12m) Also contains artifacts of the location of different detector sections. For example can light create e^-e^+ pairs at boundaries of detectors.

"pv0_x,y,z" Cartesian coordinates of primary vertex "v0_x,y,z" Cartesian coordinates of the vertex of the v0 particle

"v0_chi2" is the χ^2 under the hypothesis that two tracks stems from the same vertex. (Is calculated with covariance matrix which possibly leaves 5-7 degress of freedom.)

"v0_qOverP" is curvature of the track (?).

"thetastar" not so important, but it's the angle between a Ks and, say, a Π^+ in the Ks's reference frame. Tells something about the spin of Ks since it asks if the Π^+ has some preference for which angle it's 'spit out at'.

"rapidity/pseudorapidity" is the transformation of the angle of the Ks to perpendicular on beam direction. ie. a rapidity of 0 is perpendicular on the beam and results in more precise

measurements, whereas as rapidity of -2.5 or 2.5 results in worse measurements.

2.4.2 Correlation grouping

To generate the optimal decorrelated groups we want to create a correlation matrix and set in a block diagonal matrix. The choice of correlation metric is the Maximal Information Coefficient described earlier in section **??**. Calculating the pairwise MIC-score now gives a matrix which could be set in block diagonal form by manually changing the order of features. However, in this project the order was found by using a method of Hierarchical Clustering [**?**].

Hierarchical clustering works by initially having each point in its own cluster. The algorithm now merges two points with the shortest distance. We used $1 - \text{MIC}$ as a distance matrix and grouped according to the complete distance between two groups. Now the algorithm iterates groups merging points and repeating the grouping according to the criterion.

In the SciPy implementation of hierarchical clustering [**?**] used for this project it is possible to save the optimal ordering of the features which was saved as the ordering of parameters. Now updating the correlation matrix according to the new ordering we obtain a good suggestion to a block diagonal matrix. Now it is possible to decide groupings such that the correlation between features in one group are minimally correlated with features from others. The result of this process can be seen in **??**

2.4.3 Automatised selection

- SHAP feature importance / XGBoost implementation - Selection by best AUC

3 Analysis

3.1 K-short particles / Lambda particles

3.2 Fit and estimates

- Simple cut

As we have a theoretical value for the mass, the goal is a fit in the 'ks_mass'-parameter. The preferred fit is a double-Gaussian with a single μ. A background fit is also required, where we have found a third-degree polynomial to be the most consistent. An example of a background-fit and subsequent peak-fit in both MC and data can be seen in figure 6.

- Determine mean values and variance of the peaks in mass. Come with initial cut-values

A bunch of sanity-checks were performed on the data, to see how clean and consistent it is.

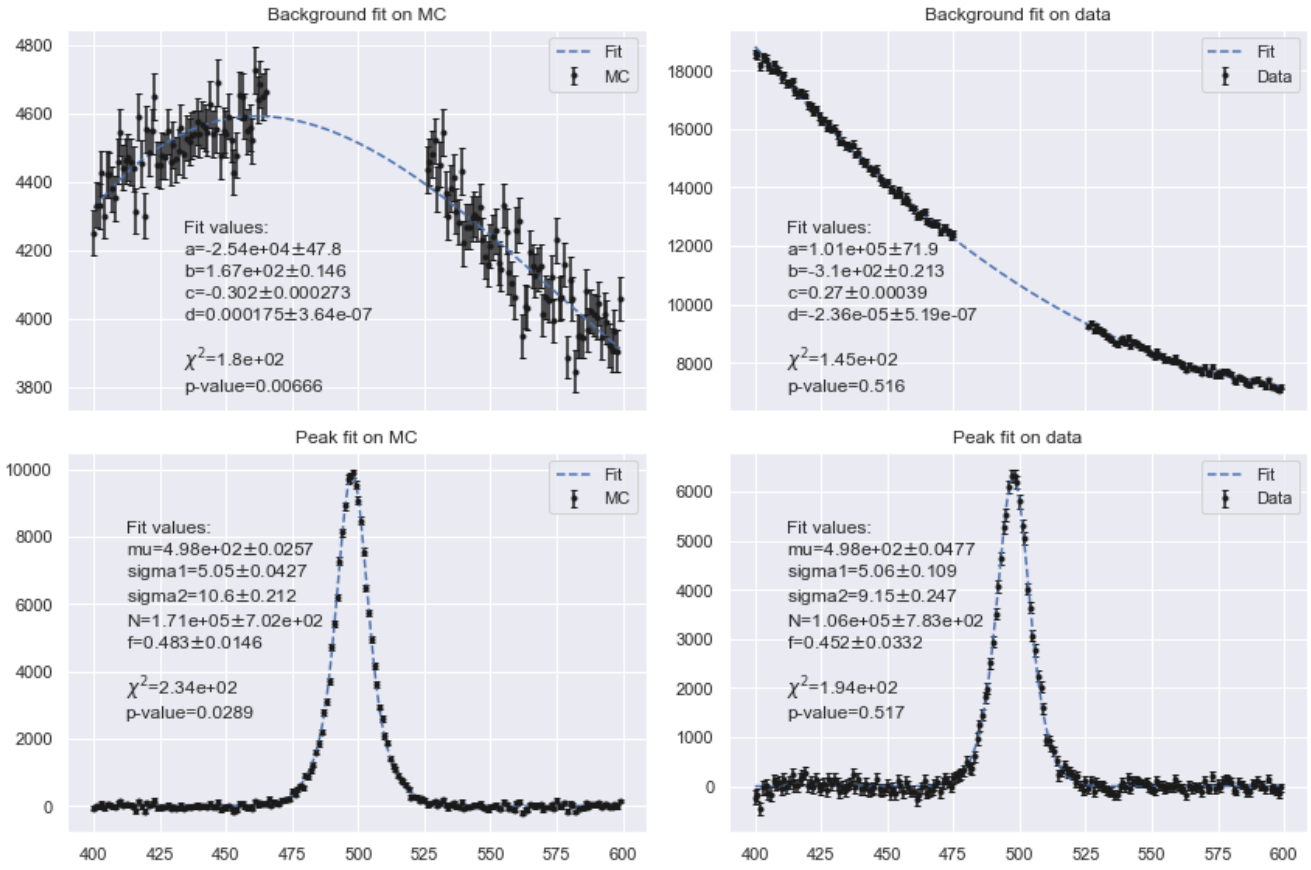

Figure 6: A typical mass-fit in untouched MC and data. The background is fitted with a third degree polynomial and a double Gaussian is used on the peak. The parameters μ, σ_1 and σ_2 are what you would expect. N is the scale parameter and f is the fraction of the scale each of the two Gaussians gets. NEED DECIMAL CHANGE

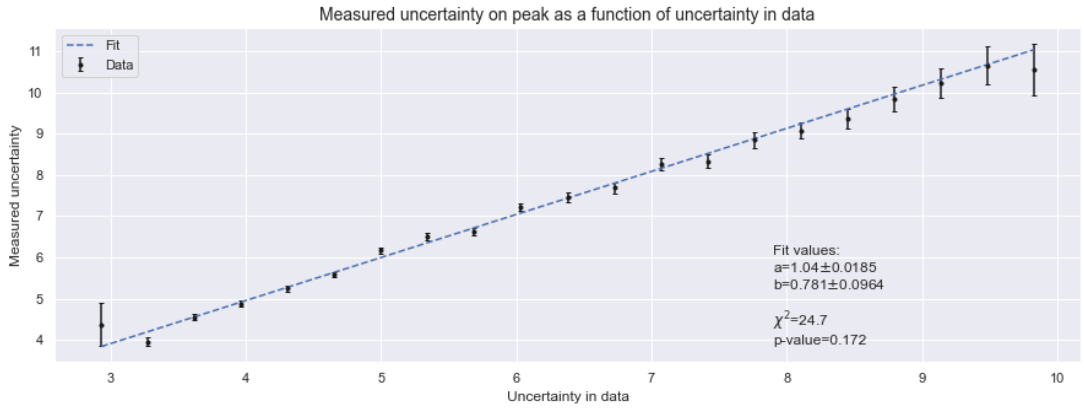

Figure 7: The correlation between the calculated uncertainty in data and the uncertainty found through fitting.

One of these consisted of picking a number of bands in the histogrammed 'v0_ks_massErr' data, and for each one try a Gaussian fit on the mass of the K-short. The variance was then extracted and can be seen in figure 7. A linear fit between the found uncertainties and the tabular values has a p-value of 0.172 giving us no reason to suspect any inconsistencies.

3.3 Parameters and correlation

- Which parameters does not correlate with mass (we want to use mass as label, this is difficult if they're correlated) - Make different uncorrelated groups

3.4 Machine Learning .¿ training and pipeline for classification

- Defining labels in data/MC-trained classification - Create a pipe line for training

3.5 Model evaluation

To evaluate models in data we need to build up tools that are robust and verifiable. In this section we will suggest a model to validate a model without using labels. The model will then be illustrated in the Monte Carlo dataset such that we can validate our method using the "trueKs"-variable. The validation method we use can be summed up in the following points:

- Using Maximal Information Coefficient as a metric for the parameters. Remove all parameters that are correlated with the mass, as we use mass to generate our labels.
- Separate parameters into two groups such that elements from one group is not correlated with parameters in the other.
- Train two models with both their set of features.

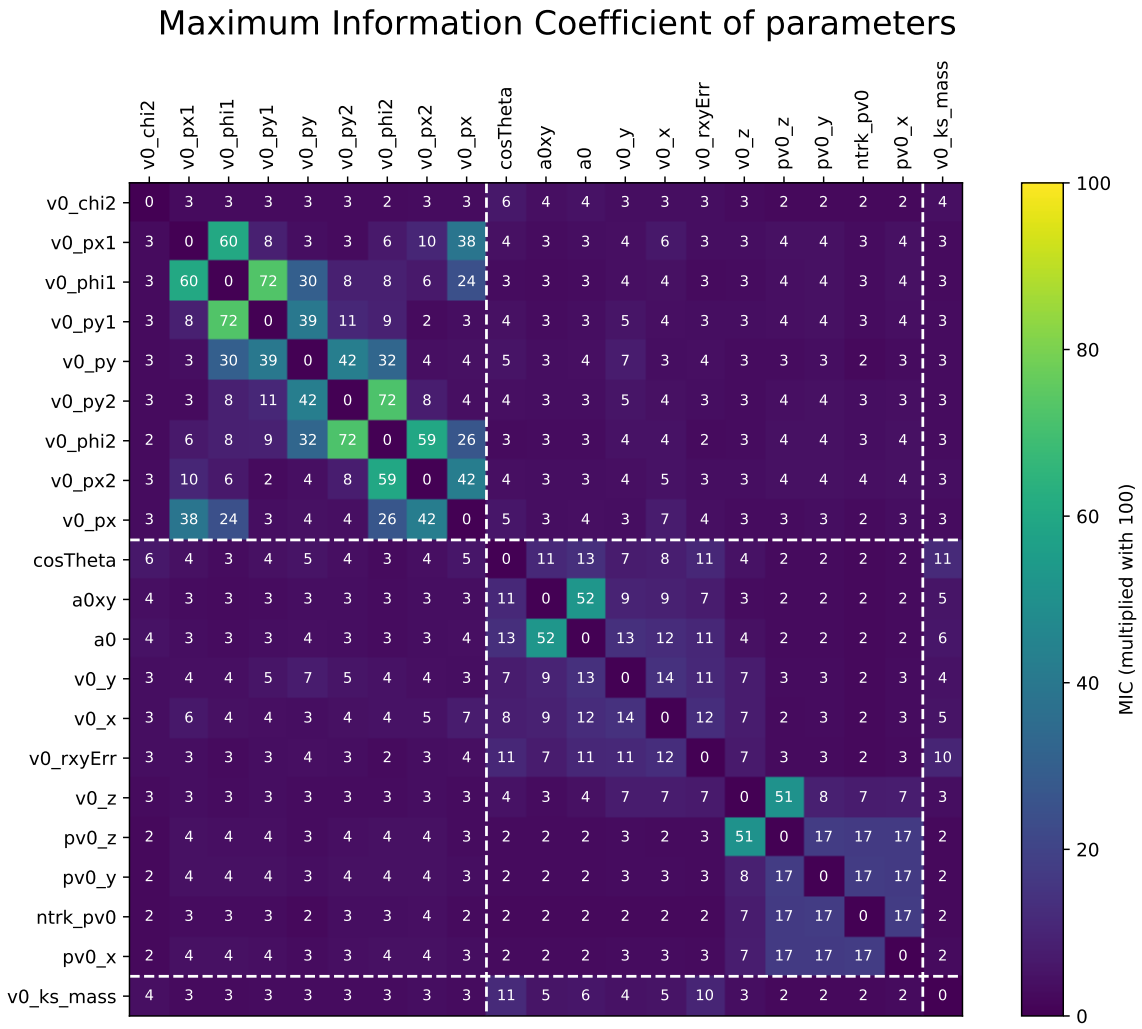

Figure 8: Overview over the MIC between parameters in the selected models. The MIC scores are multiplied with 100 to reduce clutter. The white lines indicate the two sets of parameters.

- Evaluate each model with ROC curves with the method described above. To evaluate consistency in the model
- Make ROC curves for both models using labels generated from the other method.

To test this method we tried in the Monte Carlo dataset. We decided on using the parameters illustrated in Fig 8 since it they are negligibly correlated with the mass. The parameters are now separated into two groups using hierarchial clustering with the correlation matrice treated like a distance matrix with $d(x, y) = 1 - \text{MIC}_{xy}$. We now get an optimal ordering and place the cut to minimize cross correlation between the two groups.

Now training a LightGBM (with default keywords) on a datasets in the given sets gives two models. Which can be evaluated using the method to approximate ROC curves describes in ???. Using this method gives the results displayed in Fig 9

- Perhaps something about using ML1 to create clean labels for ML2 to train on and the resulting effect on accuracy:
Even though XGBoost and LightGBM shows resilience towards erroneous labeling (atleast

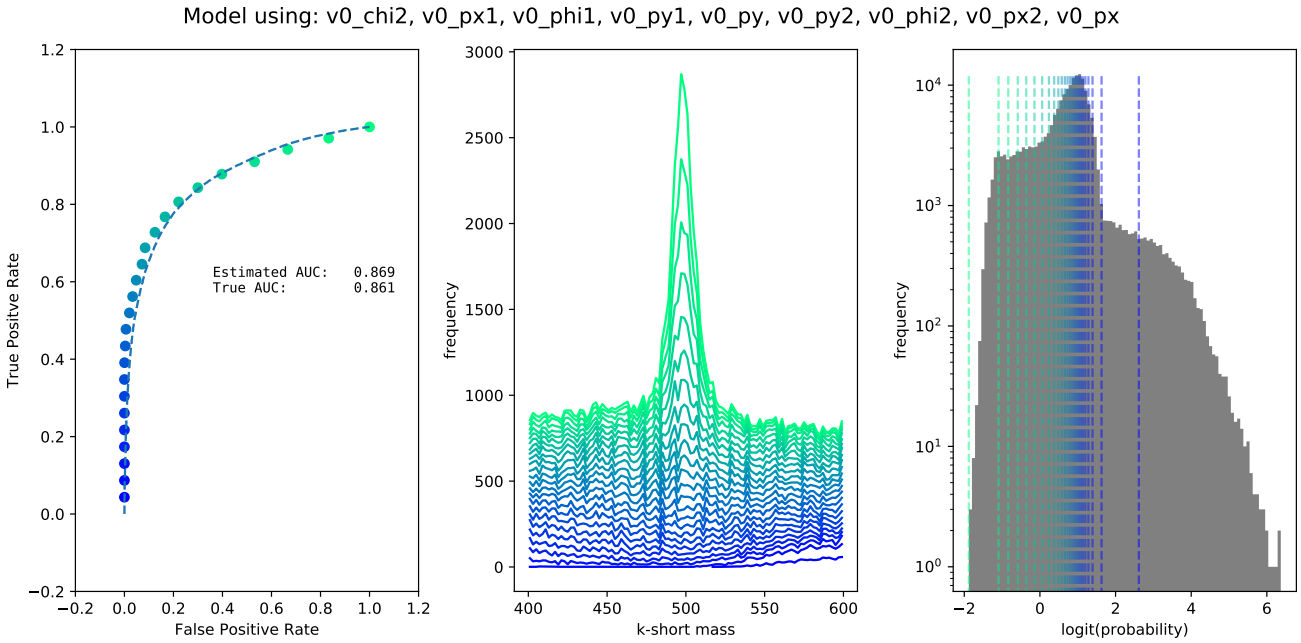

Figure 9: Models displayed ROC, logit distrubs and the histograms

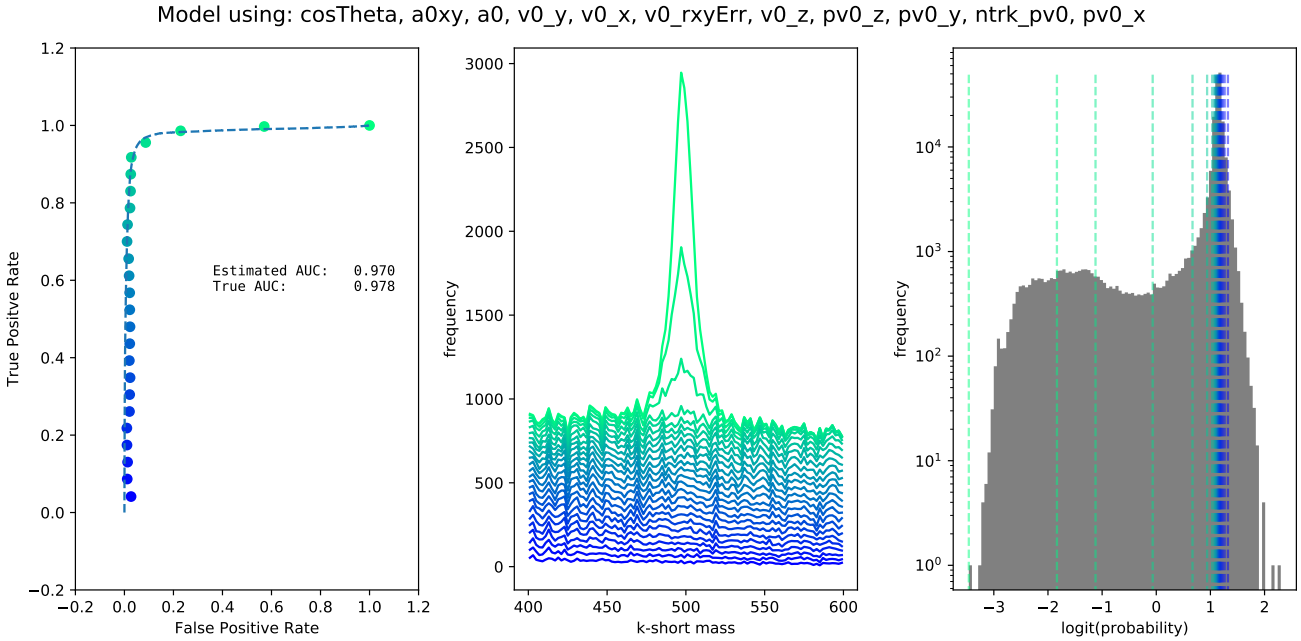

Figure 10: include in first figure

when the mis-labeling only occurs within one of the two categories[2]), it could possibly still be advantageous to reduce the ratio of mis-labeling. To do this, two set of parameters with minimal inter-pairwise correlation is selected. Then a model with the first set of parameters is trained on pseudo-labels and cuts are made in the prediction scores of new data to select data points which it is fairly confident is signal and likewise background. The second model, with the second set of parameters, is then tested on a third dataset after being trained on the second dataset with labels generated by the first model. The signal mis-labelling ratio went from $<$ 1 (~ 0.58) to $\gg 1$(~ 4.28), but whether the second model was trained on the first ratio or the second ratio had no impact, the accuracy remained the same. However, the second model was also a good model, and we can't exclude that this ratio-increase can't improve a worse model.

4 Results

$auc \approx 1$

4.1 K-short (Ks)

4.2 Lambda (Λ)

4.3 Lambda-bar ($\bar{\Lambda}$)

5 Discussion

5.1 Correlations

5.1.1 Correlations with mass

5.1.2 Cross-correlations with predictions

5.2 Accuracy

5.2.1 Choice of model

Simple cuts, ML1, Ml2 .... und so weiter

[2]With the procedure used for generating pseudo-labels, the signal category is the one that contains a lot of background as well.

6 Conclusion

See introduction

7 Appendix

Covid modelling; model prediction confidence intervals generated by use of MC.
fake uBoost? real uBoost?

References

[1] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, Advances in Neural Information Processing Systems 30, pages 4765–4774. Curran Associates, Inc., 2017.

[2] David N. Reshef, Yakir A. Reshef, Hilary K. Finucane, Sharon R. Grossman, Gilean McVean, Peter J. Turnbaugh, Eric S. Lander, Michael Mitzenmacher, and Pardis C. Sabeti. Detecting novel associations in large data sets. Science, 334(6062):1518–1524, 2011.