Diplomarbeit

Identification of muonic decays of tau pairs at the Belle II experiment through the implementation of machine learning algorithms

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Technische Physik

eingereicht von

Lukas LINAUER BSc

Matrikelnummer: 01373167

ausgeführt am Atominstitut

der Fakultät für Physik der Technischen Universität Wien

Betreuung

Betreuer: Privatdoz. Dipl.-Ing. Dr.techn. Christoph Schwanda

Mitwirkung: Dr. Gianluca Inguglia

Wien, 07.12.2020

(Unterschrift Verfasser/in) (Unterschrift Betreuer/in)

Kurzfassung

Das Belle II Experiment am Teilchenbeschleuniger SuperKEKB in Tsukuba, Japan, welcher mit der derzeit weltweit höchsten Luminosität arbeitet, soll bis zum Ende seiner Laufzeit ungefähr 50 ab^{-1} an Daten sammeln. Das sind 50 mal mehr als das Vorgänger-Experiment Belle in den 11 Jahren seines Bestehens gesammelt hat. Diese große Menge an Daten erlaubt es, die Zerfälle schwerer Leptonen mit noch nie dagewesener Präzision zu untersuchen. Speziell der myonische Zerfall von Paaren von Tau-Leptonen, $e^+e^- \to \tau^+\tau^- \to \mu^+(\overline{\nu}_\mu\nu_\tau)\mu^-(\nu_\mu\overline{\nu}_\tau)$, ist von Interesse in dieser Arbeit.

Mittels Machine Learning Methoden soll dieser Zerfall untersucht und der Wirkungsquerschnitt anhand simulierter Daten berechnet werden. Machine Learning erfreut sich zunehmender Beliebtheit in vielen wissenschaftlichen Disziplinen, nicht zuletzt wegen der stetig steigenden Rechenleistung von Verbraucher-Hardware. Ein Machine Learning Algorithmus wird anhand eines Datensatzes trainiert und lernt die vorhandenen Merkmale. Danach kann der trainierte Algorithmus auf neue, bisher unbekannte Daten angewandt werden um diese zu klassifizieren. In der Teilchenphysik kann dies verwendet werden um Signal- von Hintergrund-Prozessen zu unterscheiden. Die Hyptothese dieser Arbeit ist, dass die verwendeten Machine Learning Methoden die Daten für den untersuchten Zerfall besser unterscheiden können als dies mit klassischen Datenanalyse-Methoden möglich ist. Dazu wurden mehrere Machine Learning Methoden auf Monte-Carlo simulierten Daten trainiert und untereindander sowie mit manueller Analyse verglichen. Der Algorithmus, der die höchste Präzision auf den Test-Daten aufwies, wurde dann auf unabhängige, ebenfalls simulierte Daten angewandt, um den Wirkungsquerschnitt für die Reaktion $e^+e^- \to \tau^+\tau^-$ zu berechnen.

Es zeigt sich, dass alle der verwendeten Machine Learning Methoden eine höhere Präzision bei der Klassifikation der untersuchten Teilchenzerfälle als die klassische Datenanalyse-Methode aufweisen. Der mittels Machine Learning berechnete Wirkungsquerschnitt stimmt besser mit dem wahren Wert überein.

Abstract

The Belle II experiment, installed at the electron-positron collider SuperKEKB in Tsukuba, Japan, plans to collect around 50 ab^{-1} of data over the course of its lifetime; around 50 times more than its predecessor Belle. This presents a unique opportunity to study heavy lepton decays with unmatched precision. The aim of this thesis is to implement and test machine learning algorithms for the identification of muonic decays of tau pairs: $e^+e^- \rightarrow \tau^+\tau^- \rightarrow \mu^+(\overline{\nu}_\mu\nu_\tau)\mu^-(\nu_\mu\overline{\nu}_\tau)$.

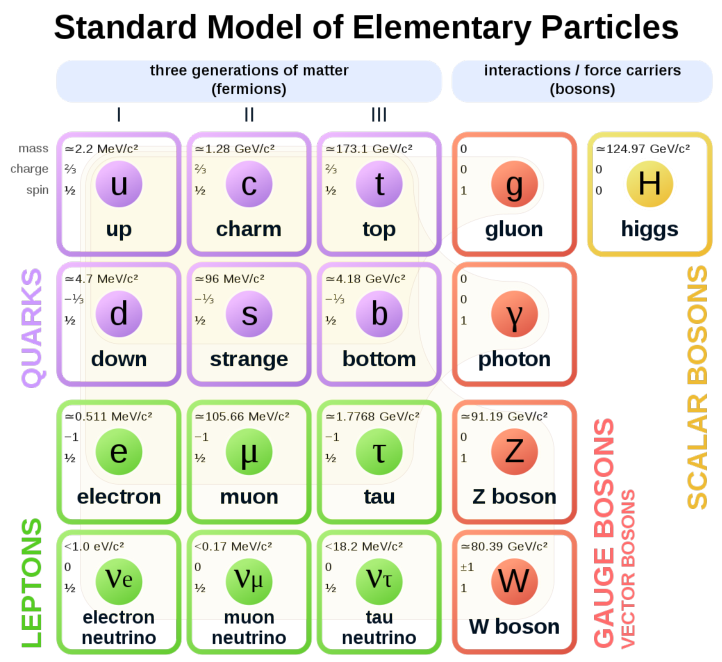
Machine Learning algorithms, which are increasingly popular in many areas of scientific research, are well suited for the analysis of large data sets. A method is trained on a set of data from which it learns to extract important features. Applied on independent data, it then tries to distinguish the signal from the background. The hypothesis of this thesis is, that these algorithms outperform humans in doing so. Different algorithms were trained on monte-carlo simulated data and then compared to one another and to classical cut-based analysis. The best performing algorithm was then used to calculate the cross-section of the process $e^+e^- \rightarrow \tau^+\tau^-$ on an independent, also simulated data set.

The results showed a superior performance of the Machine Learning models over cut-based analysis and a more accurate calculated cross-section. This suggests that these algorithms are indeed better at separating signal from background events than humans, at least in the context of the decays investigated here.

Contents

1 The Standard Model of particle physics

The Standard Model of particle physics, or Standard Model (SM) in short, is the currently accepted model of elementary particles and their interactions. Combined in the SM are three of the four currently known fundamental forces, the electromagnetic, the weak and the strong force. The underlying theoretical concept is that of quantum field theory (QFT), the combination of quantum theory with special relativity. Elementary particles can be divided into two classes, fermions and bosons. Fermions are sources of quantum fields through which they interact. Bosons are quanta of those fields. Fermions can then be further subdivided into quarks and leptons.



Standard Model of Elementary Particles
three generations of matter (fermions)
interactions / force carriers (bosons)
I
II
III
mass
charge
spin
≃2.2 MeV/c² ⅔ ½ u up
≃1.28 GeV/c² ⅔ ½ c charm
≃173.1 GeV/c² ⅔ ½ t top
0 0 1 g gluon
≃124.97 GeV/c² 0 0 H higgs
≃4.7 MeV/c² −⅓ ½ d down
≃96 MeV/c² −⅓ ½ s strange
≃4.18 GeV/c² −⅓ ½ b bottom
0 0 1 γ photon
≃0.511 MeV/c² −1 ½ e electron
≃105.66 MeV/c² −1 ½ μ muon
≃1.7768 GeV/c² −1 ½ τ tau
≃91.19 GeV/c² 0 1 Z Z boson
<1.0 eV/c² 0 ½ νe electron neutrino
<0.17 MeV/c² 0 ½ νμ muon neutrino
<18.2 MeV/c² 0 ½ ντ tau neutrino
≃80.39 GeV/c² ±1 1 W W boson
QUARKS
LEPTONS
GAUGE BOSONS
VECTOR BOSONS
SCALAR BOSONS

Figure 1: The elementary particles of the Standard Model. Credit: By MissMJ, Cush - Own work by uploader, PBS NOVA [1], Fermilab, Office of Science, United States Department of Energy, Particle Data Group, Public Domain

Figure 1 shows a list of the elemtary particles in the SM. The fermions come in three generations or flavours. Each generation is comprised of two quarks, one up-type, one down-type and two leptons. Each charged fermion has an anti-particle with equal mass but opposite charge, with exception of the neutrinos. They are mass-less and neutral particles and their anti-particles also do not have mass or charge. Fermions have half-integer spin and can be described by fermi-dirac statistics. Quarks can form bound states known as hadrons, which can be further subdivided into mesons and baryons by the number of quarks contained. A meson is formed by an even number of quarks, most simply one quark and it's anti-quark, resulting in integer spin. The lightest mesons of this kind are the pi-mesons or pions π^- , π^0 and the K-mesons or Kaons K^- , K^0. Mesons with more than two quarks are also possible and a matter of current research, e.g. tetraquarks and hexaquarks. Baryons contain an odd number of quarks, three in the simplest case and have half-integer spin. The nucleons belong to this group.

The group of bosons is comprised of the four vector bosons photon, gluon, W and Z and the Higgs boson. Each boson corresponds to one of the fundamental forces and is the carrier of this force. The photon is associated with the electromagnetic force, the gluon with the strong force and the W and Z bosons with the weak force. The Higgs boson is the quantum of the Higgs field. All elementary particles except the neutrinos gain their mass through their interaction with the Higgs field. Bosons have integer spin and follow Bose-Einstein statistics.

All particles in the SM are assumed to be point-like, a result obtained from scattering experiments. Other theories, e.g. String theory assumes that elementary particles have non-zero dimension.

Table 1 lists some of the physical quantities which are conserved for interactions in the SM.

Quantity	Symbol
Energy	E
Momentum	$\vec{p}$
Angular momentum	$\vec{L}$
Electric charge	Q
Baryon number	B
Lepton number	L

Table 1: Quantities conserved by interaction within the SM.

Each of these conserved quantities corresponds to a symmetry of the SM under a continuous transformation. E.g. the conservation of energy is a result of the symmetry under time-translations, the conservation of momentum arises due to the symmetry under space-translations. Additionally, there are the three discrete transformations, the charge, parity and time transformation. The actions of these transformations are the following: the charge transformation takes a particle into its anti-particle and thus flips the sign of the charge. Here, charge refers not only to electric charge but also to charges relevant to the strong and weak force. For a particle, represented by the function $\Psi(x, y, z, t)$, this results in:

$$C\Psi(x, y, z, t) = \overline{\Psi}(x, y, z, t) \tag{1}$$

where $\overline{\Psi}(x, y, z, t)$ is the anti-particle of $\Psi(x, y, z, t)$. The parity transformation flips the sign of all three of the spatial coordinates x,y and z.

$$P\Psi(x,y,z,t) = \Psi(-x,-y,-z,t) \tag{2}$$

It is identical to a point reflection at the origin. The time transformation flips the sign of the temporal coordinate t. It is also called time reversal.

$$T\Psi(x,y,z,t) = \Psi(x,y,z,-t) \tag{3}$$

All combinations of these transformations are valid transformations as well:

$$\begin{aligned} CP\Psi(x,y,z,t) &= \overline{\Psi}(-x,-y,-z,t) \\ CT\Psi(x,y,z,t) &= \overline{\Psi}(x,y,z,-t) \\ PT\Psi(x,y,z,t) &= \Psi(-x,-y,-z,-t) \\ CPT\Psi(x,y,z,t) &= \overline{\Psi}(-x,-y,-z,-t) \end{aligned} \tag{4}$$

In the SM, the symmetries associated with C, P and T are violated respectively. The combined transformation CP was believed long time to be a symmetry. However, already a few processes have been observed which violate this symmetry. Only the combined CPT operation is a symmetry for processes of the SM, which is known as the CPT theorem. It states, physical processes are indistinguishable when space is reflected at the origin, time is reversed and all particles are exchanged with their respective anti-particles.

$$CPT\Psi(x,y,z,t) = \overline{\Psi}(-x,-y,-z,-t) = \Psi(x,y,z,t) \tag{5}$$

The whole of QFT rests on the condition of the CPT theorem being true. Within the limits of current experimental precision, the CPT theorem could not be falsified. There are some theories however, which predict a violation of CPT symmetry.

The SM is the current best tested scientific theory but it is incomplete. Apart from the fact that it does not include gravity, there have been found a number of physical phenomena which cannot be explained by it. One example would be the neutrino oscillations, which require the existence of neutrinos with mass. In the SM however, the neutrinos have zero mass. Another example are the proposed dark matter and dark energy. Dark matter has been introduced as an explanation for the observed rotational velocities of stars in the outskirts of spiral-galaxies. These stars orbit at much higher speeds than would be expected if the masses of the galaxies are calculated by the visible matter contained in them. The missing mass is attributed to so-called dark matter. The SM also has the drawback of not being self-consistent. There are 19 free parameters, e.g. the mass of the leptons and quarks, which cannot be derived from theory alone but have to be measured in experiments. A wide variety of theories has been proposed to date, which try to enhance or extend the SM. These theories are commonly referred to as Beyond Standard Model (BSM) theories.

1.1 Lepton physics

In this section, we will explain the leptons and especially the tau lepton or simply tau in more detail. As seen in the previous section, leptons belong to the fermions and have spin 1/2. The twelve leptons are: electron (e^-), muon (μ^-), tau (τ^-), electron-neutrino (ν_e), muon-neutrino (ν_μ), tau-neutrino (μ_τ) and their respective anti-particles e^+, μ^+, τ^+, $\overline{\nu}_e$, $\overline{\nu}_\mu$, $\overline{\nu}_\tau$. The masses and mean lifetimes of the leptons are shown in table 2.

Particle	Mass [MeV]	Mean lifetime [s]
e^-	$0.5109989461 \pm 0.0000000031$	-
μ^-	$105.6583745 \pm 0.0000024$	$(2.1969811 \pm 0.000002) \cdot 10^{-6}$
τ^-	1776.86 ± 0.12	$(290.3 \pm 0.5) \cdot 10^{-15}$
ν_e, ν_μ, ν_τ	0	-

Table 2: Masses and mean lifetimes of the SM leptons according to the particle data group (PDG) [1]

The neutrinos in the SM have zero mass and are stable. Masses and mean lifetimes are identical for particle and anti-particle. The tau is about 17 times heavier than the muon and about 3550 times heavier than the electron. Its lifetime is only around one tenth of a million of that of the muon. The fact that the tau is so short-lived is also a reason for why its mass and lifetime are not nearly as accurately measured as those of the electron and muon.

The electron is the only stable lepton, apart from the SM neutrinos, being the lightest one. Muons and taus decay into lighter particles. While the muon can only decay into an electron, the tau is heavy enough to decay not only into other leptons, but also into hadrons. E.g. the decay of a tau into a pion.

$$\tau^- \to \pi^- \ \nu_\tau \tag{6}$$

As stated in table 1, the lepton number L is a conserved quantity in the SM. However, this lepton number is conserved for every lepton generation independently. One defines a lepton number for every lepton generation, for electrons, L_e, for muons, L_μ and for taus, L_τ. Leptons are assigned a lepton number of 1 and their anti-particles a lepton number of -1. If we consider the β^- decay:

$$n \to p^+ \ e^- \ \overline{\nu}_e \tag{7}$$

where a neutron decays into a proton via the emission of an electron and an electron-antineutrino, the lepton number of both sides of the arrow is zero. The lepton number L_e is conserved for the β^- decay. A hypothetical process of a muon decaying into an electron:

$$\mu^- \to e^- \, \nu_\mu \tag{8}$$

would conserve the muonic lepton number L_μ, but not the electron lepton number L_e. Such process are subject of currently ongoing research.

2 The BELLE II experiment

The BELLE II experiment is the direct successor of BELLE, which operated from 1999-2010 at the KEKB electron-positron collider located at the Japanese High-Energy Accelerator Research Organisation (KEK) in Tsukuba, Japan. KEKB is a so-called B-factory. Its center-of-mass energy is calibrated for the $\Upsilon(4\mathrm{S})$ resonance at 10.58 GeV, which nearly exclusively decays into two B mesons, hence the name. Over the course of 12 years lifetime, BELLE collected around 1 ab^{-1} of data, most of which for $\Upsilon(4S)$ but also for the other Υ resonances. In late 2010, work began to upgrade KEKB. In 2016 beam commissioning started for the new SuperKEKB accelerator. SuperKEKB is designed to work at the same center-of-mass energy as KEKB but with 40 times higher luminosity which is achieved mostly by decreasing the beam sizes at the interaction point, the so-called nanobeam scheme. Furthermore the energies of the electron beam and the positron beam have been increased from $3.5GeV$ to $4GeV$ and decreased from $8GeV$ to $7.007GeV$ respectively, leading to the same center-of-mass energy but also to a decreased boost of the center-of-mass system. The BELLE detector has been upgraded as well and will be explained in more detail in the next section. BELLE II is projected to gather around 50 ab^{-1} of data until 2025, 50 times more than its predecessor. This enables an extensive physics program which is explained in more detail in section 2.2.

2.1 The BELLE II detector

Belle II fits the same casing as BELLE and reuses some of its components, therefore major changes on the detector hull were not necessary while the performance could still be significantly increased. First data was taken in 2018 for accelerator and component commissioning. Figure 2 shows a cross section of the upper half of the detector.

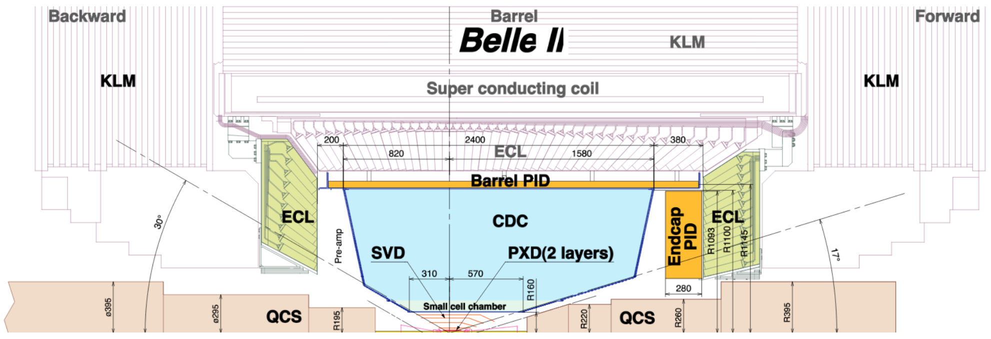


Figure 2: The BELLE II detector with its components. Shown is only the upper half of the detector, the lower half being identical. The beam pipe is located at the bottom of the picture. The backward direction is to the left, the forward direction to the right. Picture taken from [2] and slightly modified to only show BELLE II relevant parts.

The lower half of the detector is not shown but is identical to the upper half. The detector is asymmetrical in the beam pipe axis. Since the electron ring is operated at higher energy than the positron ring, the center of mass system of colliding particles is not at rest. The direction of movement of the center of mass system is called forward direction. The opposite direction is called backwards direction. The detector components from the center outwards are: Silicon Vertex Detector (SVD) and Pixel Detector (PXD), Central Drift Chamber (CDC), Particle Identification (PID) barrel and endcaps, Electromagnetic Calorimeter (ECL) barrel and endcaps and K_L^0 and muon detector (KLM). Also indicated is the super conducting solenoid magnet. All detector components except for KLM lie within the magnet. The return yoke of the magnet consists of iron plates, which are interlaced with detector plates from the KLM detector.

2.1.1 Vertex Detector

The Vertex Detector (VXD) is the innermost detector and is, as the name states, used for vertex detection. It consists of six concentric layers, each layer comprised of so called ladders. The ladders are printed circuit boards, containing both the semiconductor sensors and the read-out electronics. They are arranged symmetrically around the beam pipe. The two innermost layers are comprised of pixelated DEPFET sensors. They are located at 14 mm and 21 mm distance from the beam pipe respectively. These two layers together form the Pixel Detector PXD. It has a spatial resolution of around 10 μ m. Figure 3 shows a schematic of the PXD.

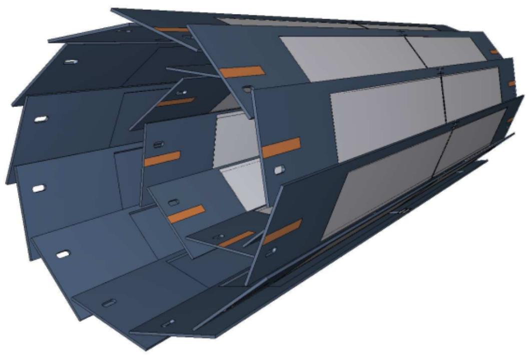


Figure 3: Schematic of PXD. Picture taken from [2]

The four outer layers comprise the Silicon Vertex Detector SVD, they are equipped with double-sided silicon strip sensors and are placed at 38 mm, 80 mm 115 mm, 140 mm, respectively. The spatial resolution of the strip sensors is lower than those of the pixel detectors. Due to close proximity to the beam pipe and the high luminosity of the accelerator, the VXD is exposed to very high particle hit rates. The material stress on the VXD is very high and the semiconductor as well as the electronics have to withstand high physical stress. High hit rates also lead to a high occupancy of the silicon strip detectors. Therefore the usage of pixel detectors in the two inner layers is crucial.

2.1.2 Central Drift Chamber

The next detector is the Central Drift Chamber (CDC). Its main functions are to reconstruct trajectories and to measure momenta $\vec{p}$ and energy loss dE/dx of charged particles. The drift chamber volume is filled with $He - C_2H_6$ gas and is riddled with 56 layers of sense wires, resulting in 14,336 sense wires in total. Charged particles traversing through the chamber volume ionize the gas. The ions then drift through the chamber and induce electromagnetic fields, which are picked up by the sense wires. In this way, the trajectory of the particles can be reconstructed. From the curvature of this trajectory, the momentum can be calculated and the change in curvature allows a measurement of the energy loss inside the drift chamber volume. CDC has a spatial resolution of about 100 μm and a resolution in dE/dx of about 11.9 %.

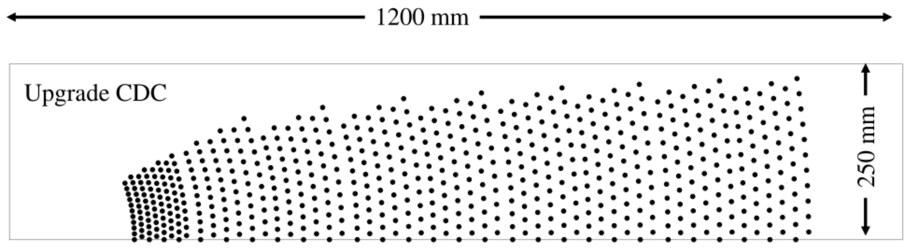
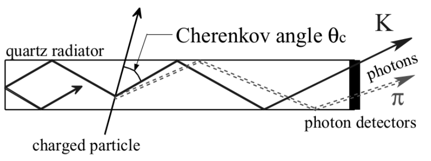


Figure 4: Wire configuration of the CDC. The dots indicate the sense wires. Picture taken from [2] and modified to only show the BELLE II relevant part.

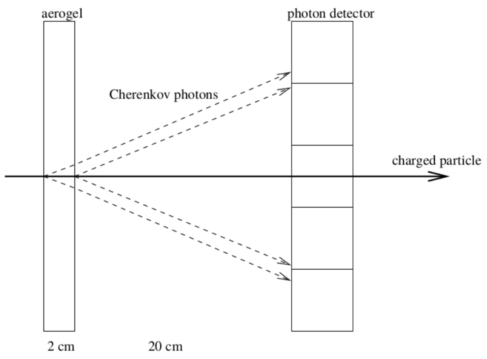
Figure 4 shows a cross section of the CDC. On the inner side of the detector, the wires are more densely placed. This is necessary to cope with high the high occupancy resulting from high hit rates on the inner part of the detector.

2.1.3 Particle Identification

The particle identification (PID) system is comprised of a barrel detector placed in radial direction and one endcap detector in the forward direction. The barrel PID system features an array of Time-Of-Propagation (TOP) counters. A TOP counter measures the time of propagation of Cherenkov photons which are produced by particles traversing a quartz radiator which are then internally reflected. The time of propagation, together with two-dimensional information from Photo-multiplier tubes allows the identification of charged particles. In the endcap, there is an Aerogel Ring Imaging Cherenkov detector (ARICH). Here, Cherenkov photons are produced when charged particles pass the aerogel radiator. The photons are picked up by a photo detector and the incidence angle is measured. The particle identity can be derived from this angle.



(a) Schematic side-view of a quartz radiator used for TOP. Shown are the path of a charged particle and the two cherenkov photons for Kaons and pions. Picture taken from [2].



(b) Schematic side-view of the ARICH detector. A charged particle traverses the aerogel radiator and produces cherenkov photons. The photon detector is actually ring-shaped. Picture taken from [2].

Figure 5: TOP and ARICH

Figure 5 illustrates the underlying principles of TOP and ARICH.

2.1.4 Electromagnetic Calorimeter

Above the PID is the electromagnetic calorimeter (ECL). There are a barrel ECL and two endcap detectors, both in forward and backward direction. The main tasks of ECL are: photon detection and photon energy measurement, electron identification and K_L^0 detection. The ECL consists of CsI crystals doped with Ti. The barrel ECL has a total of 6624 crystals, the endcaps have 1056 such crystals each. The CsI(Ti) crystals have been in use since the start of BELLE and the ECL will be upgraded in the future. There are several options for improving the reaction time of the crystals e.g. by replacing Ti doped crystals with pure CsI. At the time of writing, the old CsI(Ti) are still in use.

2.1.5 KLM

The K_L^0 and muon detector (KLM) is the outermost detector at BELLE II. Its main purposes are to identify K_L^0 mesons and muons, as the name states. There is a barrel and two endcap detectors, one in forward and one in backward direction. KLM features a layer structure, where detector layers and iron plates are interweaved. The iron plates form the return yoke for the super conducting coil and also serve as absorbers for traversing particles. The detector layers are comprised of glass-electrode resistive plate chambers (RPCs). Muon identification requires combined information of CDC and KLM. The reconstructed tracks of charged particles inside CDC are extrapolated into the KLM region and combined with information from the RPCs. K_L^0 mesons can be identified with either ECL or KLM information or with a combination of the two, leading to an increased accuracy.

2.2 Physics program

We only briefly mention the general physics program of BELLE II and mainly focus on the part concerning tau physics. The major areas of interest of BELLE II are the search for new physics (NP) in the flavour sector and the improvement of SM parameter measurements. Some questions regarding this subjects from the BELLE II Physics Book [3] are:

- Are there new CP violating phases in the quark sector?
- Does nature have multiple Higgs bosons?
- Are there sources of lepton flavour violation (LFV) beyond the SM?
- Is there a dark sector of particle physics at the same mass scale as ordinary matter?
- What is the nature of the strong force in binding hadrons?

The SM has been tested and found to be in very good agreement with experiments up to the TeV scale. High-energy physics (HEP) experiments can address the questions stated above in two ways. In the direct or energy frontier approach, particles are collided at very high energies in order to directly produce new particles. This is the approach followed by the LHC, which collides protons at center-of-mass energies of around 14 TeV. The indirect or intensity frontier approach, followed by BELLE II, strives to examine NP by measuring discrepancies with the SM. Crucial for this approach are large amounts of data, produced in high-luminosity or high-intensity colliders, which give this approach its name.

Due to the high luminosity of the SuperKEKB accelerator, tau physics can be studied at BELLE II with unprecedented precision. Around $45 \cdot 10^9$ tau pairs are expected in the full data set. Another advantage of SuperKEKB regarding tau physics, is the fact that it is an electron positron collider and therefore the production cross-section for tau pairs is comparatively high. One important phenomenon under consideration at BELLE II is the charged lepton flavour violation in tau decays. Lepton flavour violation (LFV) is not known in the SM but has already been observed in the neutrino sector in the form of neutrino oscillations. Charged LFV on the other hand has not yet been observed. LFV decays are e.g. the decays $\mu^- \to e^-\gamma$ or $\tau^- \to \mu^-\gamma$. Decays involving μ have already been examined and stringent bounds found for the branching ratios. LFV decays including τ have not been as thoroughly tested. Especially the decays $\tau^- \to \mu^-\mu^+\mu^-$ and $\tau^- \to \mu^-\gamma$ are promising candidates for the study of charged LFV.

In the SM, the Cabibbo-Kobayashi-Maskawa (CKM) matrix describes the mixing of quarks of the three flavours. It can be described by three real parameters and one complex phase. The physical importance of this complex phase lies in the CP violation of electro-weak decays i.e. decays which involve the electromagnetic as well as the weak force. This CP violation affects quarks, but not leptons. CP violation in the lepton sector is subject to current research. At BELLE II, semi-hadronic tau decays, i.e. decays which have hadrons and other particles in the final state like $\tau^+ \to \pi^+ K_S^0 \overline{\nu}_\tau$, are of special interest for this matter. If CP violation is found in one of these decays, it is an indication of an existence of a CP violating processes outside of the CKM mechanism.

2.3 BASF2

The Belle 2 Analysis Framework (BASF2), is the main software framework for simulation, reconstruction, visualization and analysis for BELLE II. There are around 40 packages, e.g. one for each detector, a package for reconstruction, one for analysis, etc. These packages are written in C++. BASF2 is available on central servers like the DESY NAF or at the KEK Computing Center. It is also possible to install BASF2 locally on a machine, either by building the source code or via a Docker Image. BASF2 has an extensive Python interface. Python is an increasingly popular programming language. It is designed for readable code and straightforward software design and is widely used for Data Science applications. Users can write Python scripts, so-called steering files and execute them via BASF2. A typical analysis workflow combines different modules to a chain, which is sequentially processed. In BASF2, such a chain of modules is called path. Figure 6 shows a sample path in BASF2.

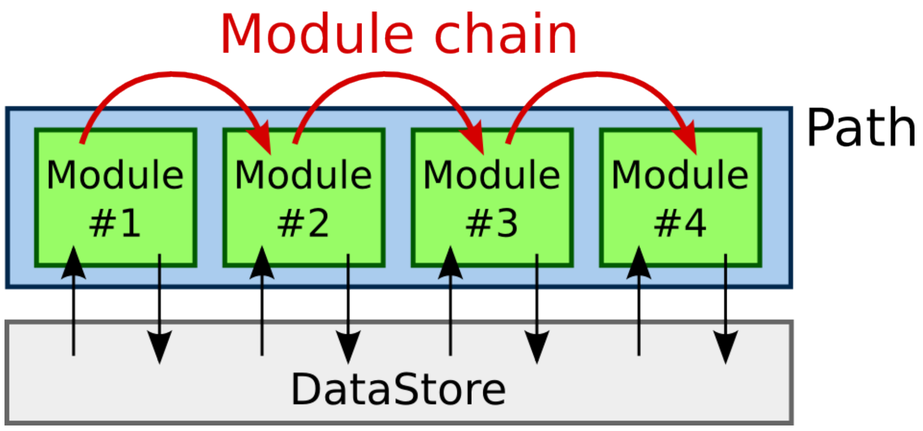


Figure 6: Modules are linked together to form a path. Picture taken from [4].

Each module in the path can have access to data. The path processing is strict, only when a module has finished successfully, the next will start processing. Besides steering files, users can work with BASF2 as a Python interpreter. Based on the Ipython interpreter, it features functionalities like tab completion, system shell access and history retrieval. With this interpreter, it is possible to work with data interactively or to test modules. This interpreter can also be used inside Jupyter Notebooks. Jupyter is a web-based interactive environment which lets users run code in notebooks. A notebook consist of a list of cells which can be used to run code or display text or images. Besides interactive access, Jupyter Notebooks also have the advantage of an internal documentation. Notebook can be saved and re-executed, whereas the command line interpreter does not save progress. Along with BASF2, it is also possible to use ROOT inside a Jupyter Notebook.

A module which is heavily used in this thesis is called BASF2_MVA. It features software for methods of multivariate analysis, most notably Machine Learning. The module provides an interface, which is written for C++, bash and Python, for several backends, e.g. TensorFlow and scikit-learn. The standard machine learning backend is called FastBDT. It is an implementation of a boosted decision tree classifier specifically written for the use in the BASF2 framework. The main goal of the BASF2_MVA module is to work backend-independent. Functions like, training, validation and inference should be applicable to all available backends, which also enables it to compare methods from different backends.

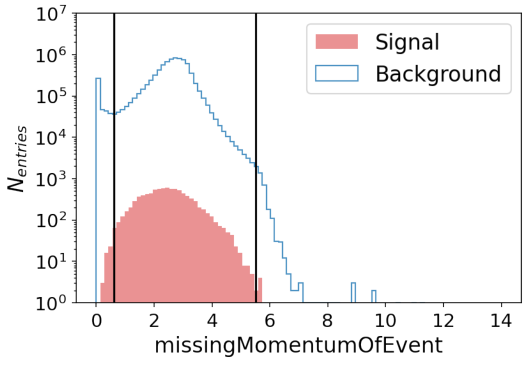
3 Machine Learning

In this section, an introduction to machine learning is given together with a closer look at some classification algorithms.

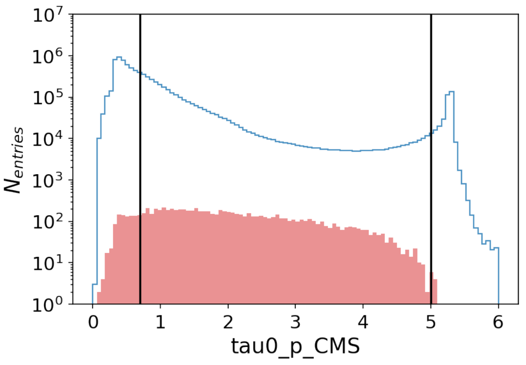
In HEP, what one often wants, is the classification of certain data points as belonging to one of two classes, signal and background. Signal events are all those, which belong to the process of interest, background events all those who don't. A very simple, yet widely used and powerful way of classification is by rectangular cuts.

A cut is a simple restriction of the domain of a variable. For example, if the variable $x \in \mathbb{R}$ is restricted to values $x > 0, x < 10$, we speak of a cut. The systematic approach for a cut-based analysis looks like the following. First, simulated data is examined. Distributions of relevant variables are plotted for signal and background events. Then, cuts are applied to each variable so that the number of background events which are dismissed is maximized while at the same time the number of signal events dismissed is minimized. The ordering in which the cuts are applied is relevant. If a cut has no marginal effect, that is, if no events are ruled out by this cut after all other cuts have been applied, it should not be included. The marginal effects of every cut should be examined. Then, after all cuts have been applied, one counts the remaining signal and background events and calculates the quantities of interest. After the analysis has been optimized on simulated data, one can proceed to apply it on real data.

Figure 7 shows two example variable distributions and the corresponding cuts.



(a) Example histogram of the distribution of the variable missingMomentumOfEvent.



(b) Example histogram of the distribution of the variable tau0_p_CMS.

Figure 7: Two distributions of variables for the process $e^+e^- \to \tau^+\tau^- \to \mu^+\mu^-$ (signal) and background processes. The vertical black lines indicate where the variable was cut.

All events inside the cut region are treated as signal events, all events outside as background events. We speak of rectangular cuts, because the cut chooses a rectangular region in the one dimensional distribution of a variable. It is also possible to cut on two dimensional distributions of variables and therefore make use of possible correlations of these variables. Manual cutting in higher-dimensional regions is difficult. Even the depiction of distributions in more than two dimensions is not straightforward, let alone finding the optimal choice of cuts. This is a severe limitation of the cut-based approach and machine learning methods provide a way of dealing with this issue.

Machine learning or statistical learning describes a collection of algorithms which aim to learn certain features from data to build up a model and then infer this model on unseen data. To formalize this concept, consider the functional relation

$$y = f(\vec{x}), \quad \vec{x} \in \mathbb{R}^n, y \in \mathbb{R}, \mathbb{N} \tag{9}$$

where $\vec{x} = (x_1, x_2, ..., x_n)^T$ are called independent or input variables and y is called the dependent or output variable. The goal is to find a function

$$\hat{y} = \hat{f}(\vec{x}) \tag{10}$$

which approximates y. To achieve this, we first have to make assumptions on the form of $\hat{f}$. There are a variety of possibilities for the choice of $\hat{f}$ which define the learning algorithm. For example, $\hat{f}$ can be a linear or a polynomial function, but also more sophisticated models, like non-parametric or additive functions are possible. The next step is to look at labeled pairs of data $(\vec{x}_1, y_1), (\vec{x}_2, y_2), ...,$ evaluate the predictions $\hat{y}$ and calculate the error or loss function $E(y, \hat{y})$, which is a measure of the deviation of $\hat{y}$ from the actual y. The model is then updated to minimize this error function. After this learning or training phase, the model is applied to a test data set and the error again calculated. The correctness of the model can be assessed with this error. Data from which the model learns is called training data, data on which the model performance is evaluated is called test or validation data. After the model has been trained, it can be used to predict properties on unseen data. This procedure is called supervised learning, as the training data has to be correctly labeled. There are also methods for unsupervised learning, where no labels are supported or a mixture of both, where only some data points are labeled. We will focus here only on supervised learning methods. The described process is very general and there are many different algorithms, which differ in the form of $\hat{f}$, which error function $E(y, \hat{y})$ to use and the domain of the output variable y.

Two classes of machine learning algorithms, regression and classification algorithms can be distinguished. We speak of regression algorithms if $y \in \mathbb{R}$ and of classification if $y \in \mathbb{N}$. We restrict ourselves to classification algorithms in the following sections.

3.1 Classification algorithms

A classification problem is such that all data points can be identified as belonging to one of several classes, i.e. $y \in \{0, 1, 2, 3 \ldots N\}$. In the easiest case, and the one we are interested in here, only two classes are present, e.g. $y \in \{Signal, Background\}$. For every data point, the algorithm has to decide which class it belongs to.

3.2 Logistic Regression

The simplest classification algorithm is the logistic regression. It can separate two classes with a linear decision boundary. The name logistic regression stems from the logistic function

$$\begin{aligned} f &: \mathbb{R} \to (0, 1) \\ f(x) &= \frac{1}{1 + e^{-x}} \end{aligned} \tag{11}$$

also known as sigmoid curve, which is shown in figure 8. Despite the name, logistic regression is actually a method for classification.

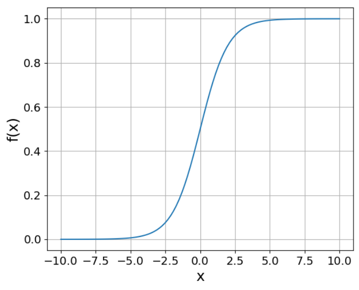


Figure 8: Logistic function in the range $x \in [-10, 10]$

The logistic function takes on values between 0 and 1, so we can interpret its output as the probability $p(x)$ of a data point x belonging to one of two classes. The probability of x belonging to the other class is then $1 - p(x)$. Logistic regression therefore can only be used for distinguishing between two classes. For data with more than one input variables, we form the linear combination of the inputs

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots \tag{12}$$

with the parameters β_i. These parameters, also called weights, are to be estimated from the training data. The probabilities of an input $\vec{x}$ are then calculated by applying the logistic function to this linear combination

$$p(\vec{x}) = f(z) = \frac{1}{1 + e^{\beta_0 + \beta_1 x_1 + \ldots}} \tag{13}$$

In practice, one sets a critical value and treats all observations which result in probabilities higher than this value as belonging to class 1 and all other as belonging to class 2. Although the logistic function is non-linear, because one calculates the linear combination of input values, the resulting decision boundary is indeed linear. The decision boundary is the hyperplane in the input variable space, which separates class 1 from class 2. In two dimensions, this is just a straight line. Figure 9 illustrates this for two input variables x_1 and x_2.

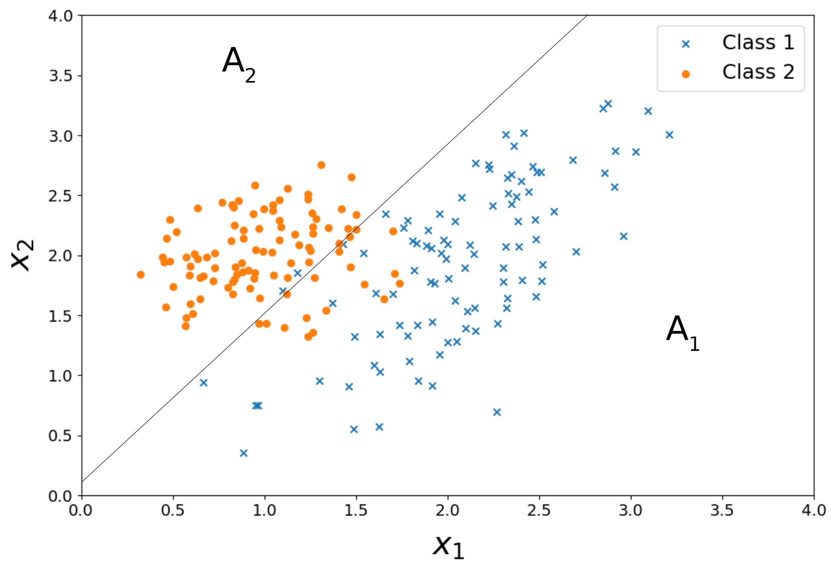


Figure 9: The decision boundary from logistic regression in two dimensions is a straight line. The markers indicate to which class each data point actually belongs.

The line separates the variable space into two distinct regions A_1 and A_2. All points to the right of the decision boundary would be assigned to class 1 and all points to the left to class 2. In this example, quite a few data points from class 2 would be wrongly classified as belonging to class 1. This indicates, that a straight line cannot separate the two classes perfectly and logistic regression is too simple for this problem.

3.3 Decision Trees

Decision trees are a general method for decision making. A generic example of a decision tree is depicted in figure 10.

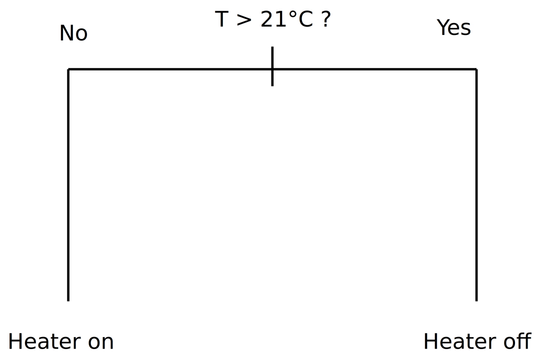


Figure 10: Example of a decision tree for a heating control system.

Shown is a tree for a simple heating control system. If the temperature surpasses the defined threshold of 21 degrees Celsius, the heater will be turned on. If the temperature falls below this value, the heater will be turned off. This tree has only one split and therefore a depth of one. Decision trees can be of arbitrary complexity, depending on the problem. Splits in the tree are called branches and terminal nodes are called leafs. The rectangular cuts explained above are equivalent to trees of depth one. Decision trees can be used for both classification and regression. The procedure of training a classification tree on data looks as follows:

First, we split the input variable space of the training data into two regions $x_1 < c_1$ and $x_1 > c_1$ and calculate the Gini index

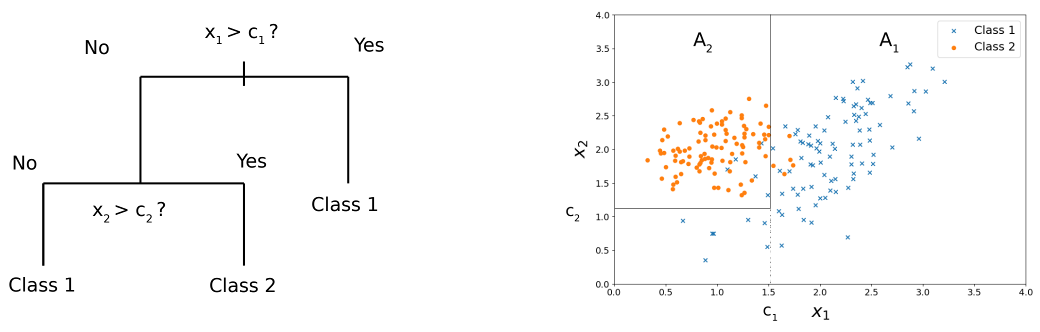
$$G = \sum_{k=1}^{K} \hat{p}_k (1 - \hat{p}_k) \tag{14}$$

in both regions. Here $\hat{p}_k$ is the proportion of data points which belong to the class k in a region and K is the total number of classes. The Gini index is close to zero, if many data points are correctly classified. Therefore it is also called node purity. The optimal value for c_1 is such that it minimizes the sum of the Gini indices of both regions. For the next step, split these regions further using the same measure to calculate the optimal split values. We stop when every terminal node has reached a certain purity. This process is called recursive binary splitting. Another measure for calculating the optimal split values is the cross-entropy D

$$D = -\sum_{k=1}^{K} \hat{p}_k log(\hat{p}_k) \tag{15}$$

Like the Gini index, the cross-entropy takes on small values if many data points are correctly classified. If the number of possible classes $K = 2$, then we speak of binary cross-entropy.

With a grown tree, the classification is straightforward. Figure 11 illustrates this procedure.



(a) A decision tree with two splits and three leave nodes

(b) The solid line indicates the final decision boundary. The dashed line indicates the first split.

Figure 11: Classification with a decision tree.

The data set is the same as in the logistic regression example. Starting at the top of the decision tree in figure 11(a), the data is split into two regions $x_1 > c_1$ and $x_1 < c_1$ at the first branch. Data points with $x_1 > c_1$ are assigned to class 1. The second branch splits the region with $x_1 < c_1$ further. Data points with $x_2 > c_2$ are assigned to class 2 and those with $x_2 < c_2$ to class 1. The result is two disjoint regions A_1 and A_2 with a highly nonlinear boundary. Graphically, all data points which lie inside region A_1 are assigned to class 1, and vice versa. The resulting decision boundary better separates the two classes than in the case of logistic regression. The decision boundaries of classification trees are straight line pieces joined together at right angles. This case is the other extreme to the linear decision boundary of logistic regression. Growing a single tree is often vastly inferior to many other classification methods. To improve the performance, one can use a method called boosting.

3.3.1 Boosting

Boosting is a method of improving the accuracy of any statistical learning method and not just limited to the use in decision trees. The main idea of boosting is to subsequently train identical methods on modifications of the same data and average the predictions. The goal is to improve the predictions that one method alone would give.

Here, we give an overview of how boosting works with classification trees. First, scale the training data with weights $w_i = \frac{1}{N}, i = 1, 2, ..., N$, where N is the number of data samples. Train a single tree of depth d on this scaled data. For $d = 1$, the tree only has a single split. This classifier is called T_1. Increase the weights for data points which were wrongly classified by T_1 and decrease the weights for correctly classified ones. The rate of de-/increase depends on the number of wrongly classified and a parameter λ called learning rate. Train a new classifier T_2 on the scaled data and update the weights again. Repeat this procedure K times. The probability of an input $\vec{x}_j$ belonging to a class is then

$$p(\vec{x}_j) = \sum_{i=1}^{M} \lambda T_i(\vec{x}_j) \tag{16}$$

where $T_i(\vec{x_j})$ is the probability prediction of the i-th classifier. The probability of belonging to the other class is again $1 - p(\vec{x_j})$. This algorithm is known as AdaBoost [10]. The advantage of boosting is that the resulting method learns "slowly", meaning that it gradually improves the predictions with each iteration. This approach is less susceptible to over-fitting.

Boosted decision trees (BDT) have three hyper-parameters: the number of iterations M, the learning rate λ and the depth of the trees d. While $d = 1$ is often a good choice for boosting, it can be difficult to find good values for λ and M, since they are interdependent. Decreasing λ can lead to better performance but requires a larger M which in turn can lead to over-fitting the data. The optimal set of hyper-parameters can be found with cross-validation. It is also possible to calculate a specific learning rate λ_i for each iteration. Often however, a constant λ produces good results.

Hyper-parameters and cross-validation will be explained in more detail in section 3.6.

3.4 Gaussian Naive Bayes classifier

The Gaussian Naive Bayes (GNB) classifier is a special form of Naive Bayes (NB) classifier. NB classifiers are a family of methods which "naively" assume conditional independence between every pair of data points (x_i, x_j) given the class label y.

$$P(x_i|y, x_1, x_2, \dots x_{i-1}, x_{i+1}, \dots x_n) = P(x_i|y), \ \forall i \tag{17}$$

With this assumption, we can classify events according to the following scheme. Starting from Bayes rule for a set of input data points $x_1, \dots, x_n$

$$P(y|x_i, \dots x_n) = \frac{P(y)P(x_1, \dots, x_n|y)}{P(x_1, \dots x_n)} \tag{18}$$

The joint conditional probability $P(x_1, \dots, x_n|y)$ can be written as

$$\begin{aligned} &P(x_1|y, x_2, \dots x_n) \cdot P(x_2|y, x_3, \dots x_n) \cdots P(x_n|y) \\ =&P(x_1|y) \cdot P(x_2|y) \cdots P(x_n|y) \\ =&\prod_{i=1}^{n} P(x_i|y) \end{aligned} \tag{19}$$

using the conditional independence assumption. We then can rewrite Bayes rule accordingly:

$$P(y|x_i, \dots x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i|y)}{P(x_1, \dots x_n)} \tag{20}$$

$P(x_1, \dots x_n)$ is constant for given $x_1, \dots, x_n$ i.e. for the training data. This constant can be neglected for the calculation of the predicted class label. The predicted class label is then calculated as

$$\hat{y} = arg \underset{y}{max} \left(P(y) \prod_{i=1}^{n} P(x_i|y) \right) \tag{21}$$

The predicted class label is the value of y, which maximizes $P(y) \prod_{i=1}^{n} P(x_i|y)$. Now, both $P(y)$ and $P(x_i|y)$ are unknown, so further assumptions have to be made. For training data with known class labels, we can assume $P(y)$ to be the relative frequency of the class label in the training data set. For the conditional probability $P(x_i|y)$, we can make different assumptions which serve as a distinction between different NB classifiers. For the GNB, we assume $P(x_i|y)$ to be

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2} \right) \tag{22}$$

The parameters μ_y and σ_y^2 are estimated using the maximum likelihood method. The GNB classifier performs relatively well while being reasonably fast during training.

3.5 Artificial Neural Networks

Artificial Neural Networks (ANNs) are inspired by and somewhat modeled after the human brain. An artificial neuron, like a real neuron can receive input signals which when strong enough cause the neuron to "fire", meaning to produce an output signal. A number of such neurons can be combined in a network and used for learning purposes. There are a wide variety of ANNs used for such purposes as computer vision, natural language processing or reinforcement learning. These are typical use cases for deep learning and in fact the name deep learning stems from the usage of deep neural networks, meaning neural networks with many layers of neurons. However it is also possible to use ANNs for simple two-class classification problems. This section shows the learning process of artificial neurons and how these neurons can be combined to form ANNs.

A single artificial neuron can be thought of as a function

$$\begin{aligned} &f : \mathbb{R}^n \to \mathbb{R}, \mathbb{N} \\ &f(x) = \phi(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + ...) \end{aligned} \tag{23}$$

where ϕ is called the activation function of the neuron. The neuron takes the weighted sum of the inputs and applies to it the activation function. There are many different choices for ϕ. Two prominent ones are the Heaviside theta function $\Theta(x)$ or the identical function $id(x) = x$. The choice $\phi(x) = \Theta(x)$ is called Perceptron, while the neuron with $\phi(x) = id(x)$ is called adaptive linear neuron or Adaline. In the case of the Perceptron, the output of the activation function is either 0 or 1, so it can only be used for distinction of two classes. The Adaline may be used for regression as well as classification. In the latter case, one can interpret the output as the probability for class membership.

In order for an artificial neuron to learn, we need to specify a loss function $E(y, \hat{y})$. For the Adaline, we choose the loss function

$$E(y, \hat{y}) = \frac{1}{2} \sum_{i=1}^{N} \left(y^{(i)} - \phi(\vec{x}^{(i)}) \right)^2 \tag{24}$$

the mean squared error. This loss can be expressed as a function of the weights β_i, $E(y, \hat{y}) = L(\vec{\beta})$, where $\vec{\beta} = (\beta_0, \beta_1, ..., \beta_N)^T$. The goal is to minimize the loss function. This is done by gradient descent.

$$\vec{\beta}' = \vec{\beta} - \lambda \nabla L(\vec{\beta}) \tag{25}$$

The weights are updated with the gradient of the loss function multiplied with the learning rate λ and the neuron trained again. This is done in an iterative way, until the loss function reaches a minimum. By specifying the activation function, the loss function and initializing the weights, the artificial neuron is completely defined. It is a simple and easily understandable model. The disadvantage however is the relative weak performance in comparison with other methods, especially when facing many input variables. To alleviate this, we can combine single neurons to an ANN.

3.5.1 Multilayer Perceptron

A simple form of ANN is called the multilayer Perceptron (MLP). It consists of multiple layers of neurons, the input layer, the output layer and one or more hidden layers. The input layer contains one neuron for every input variable, the output layer has $N_{classes} - 1$ neurons. The hidden layers lie between input and output and are therefore isolated from the outside, hence the name. How many hidden layers and how many neurons in each layer depends strongly on the given problem. A good starting point for a binary classification problem is to just use one hidden layer with $(N_{input} + N_{output})/2$ neurons. Every neuron in one layer is connected to every neuron in the following layer. Figure 12 shows an example of a MLP.

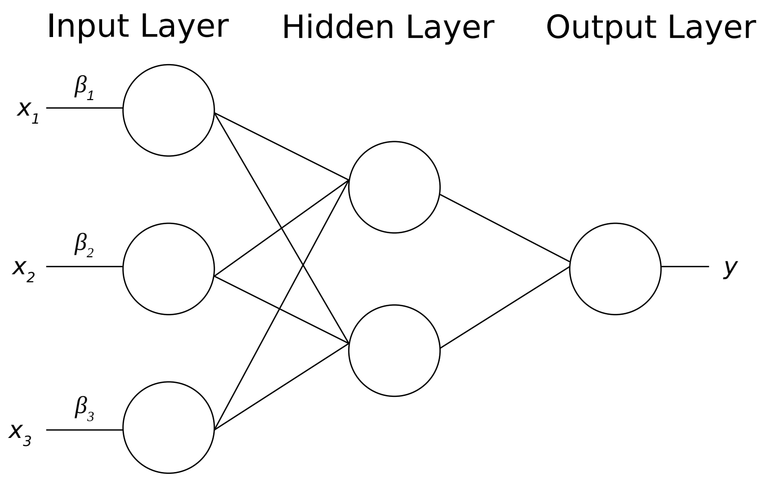


Figure 12: Example structure of a MLP with three input neurons, one output neuron and one hidden layer with two neurons. The weights between the input and hidden layer and between the hidden and output layer are omitted.

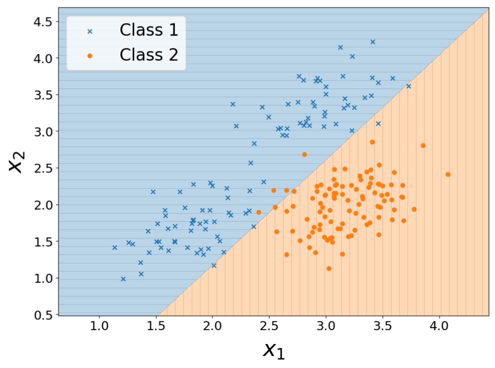
The input layer neurons only have one input, the input variables x_i. Figure 12 is an example of a fully-connected neural network with identical neurons. Every neuron in one layer is connected to every neuron in the next layer, with each neuron having the same activation function. Networks which are not fully-connected or which feature multiple types of neurons are also possible. The learning procedure of a ANN is similar to that of a single neuron. After a loss function is specified and all weights are initialized, the data is fed to the network. For every data point in the training set, each input is assigned to one input layer neuron. Each neuron applies the activation function to its input and transmits the output to the neurons of the next layer. This is done until the output layer is reached, where the loss is calculated. Via gradient descent, the weights are updated beginning from the output layer neurons recursively until all weights are updated. This method is called back-propagation. When the number of training samples becomes large, the gradient descent method becomes infeasible to calculate. In order to cope with this, only a subset of the training data is used for calculating the gradient of the loss function. The true gradient is then estimated and the weights are updated accordingly. This method is called stochastic gradient descent (SGD) and is the basis of all modern deep learning algorithms.

3.6 Parameter tuning and validation

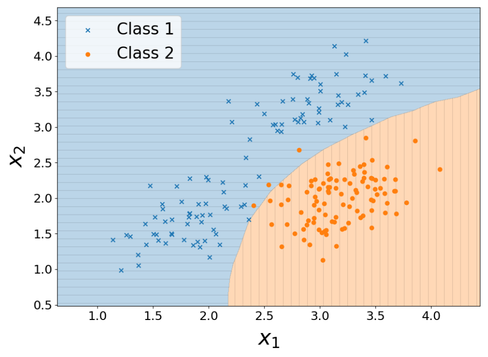
Choosing the right model for a problem is an important step in the process. The model has to be just complex enough to be appropriate for the data at hand, so that predictions are accurate. If one chooses a too simple or too complex model, the prediction error increases. This is known as the bias-variance tradeoff. The terms bias and variance are tied with the concepts of over- and underfitting. Underfitting means trying to fit a distribution of data with a too simple model where it will not be possible to model the true distribution of data correctly. The resulting error is called bias. Overfitting occurs when the model can fit the training data too good. If the data would change only a little, the model prediction would decrease dramatically. This is called variance. An example of the effect of model complexity on two-dimensional data is shown in figure 13. Three different models with increasing complexity are trained and their predictions plotted together with the actual class affiliation.

The logistic regression leads to a linear decision boundary, which is too inflexible for the shown data. Bias is likely to be high in that case. Quadratic Discriminant Analysis results in a parabolic decision boundary, which appears to be adequate for this problem. The 1-Nearest-Neighbour Classifier results in a very curvy decision boundary. While for this problem, it correctly classifies all data points, the accuracy would decrease greatly if the data points were to be altered only a little. The accuracy of a classifier can be measured by the training and test errors. The training error is the proportion of correctly classified events in the training data set. The test error is the same measure calculated for the test or validation data set. While the training error can be reduced indefinitely with increasing model complexity, the test error will vary non-linearly. High bias as well as high variance will lead to large test errors with a minimum in between, resulting in a u-shaped curve depicted for the example of a K-Nearest-Neighbour Classifier (KNN) in figure 14. This u-shaped curve for the test error appears for all machine learning methods and illustrates the problem of the bias-variance tradeoff.

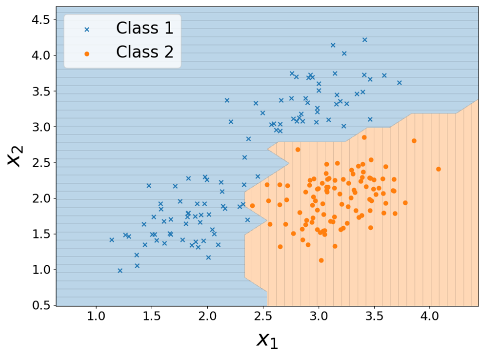
To minimize the test error, model complexity has to be carefully selected. This can be done via hyper-parameters. The model parameters, or weights, are estimated during the learning process. Hyper-parameters cannot be chosen from data alone, one has to select



(a) Logistic Regression



(b) Quadratic Discriminant analysis



(c) 1 Nearest Neighbour Classifier

Figure 13: Visualization of over- and underfitting. Three models with different complexity are trained on the same data. Markers indicate the actual class of the data points and the shaded region correspond to the predicted classes. All points inside the horizontally hatched region is assigned to class 1, all points inside the vertically hatched regions to class 2.

them either manually or by validation. Examples of hyper-parameters are the number and depth of trees in BDTs, the learning rate and the number of neurons in ANNs. One way of determining hyper-parameters is cross-validation.

The training data is split up into m subsets. The model is trained on $m - 1$ of those subsets and the test error evaluated on the remaining one. This process is repeated m times and then all test errors are averaged. This average is used as a measure to assess the hyper-parameters. We speak of m-fold cross-validation. In the case where m is equal to N, the number of data points in the set, the process is called Leave-One-Out cross-validation. Depending on the size of the data set, performing cross-validation can be very compute intensive. For very large data sets, it is also possible to do the validation

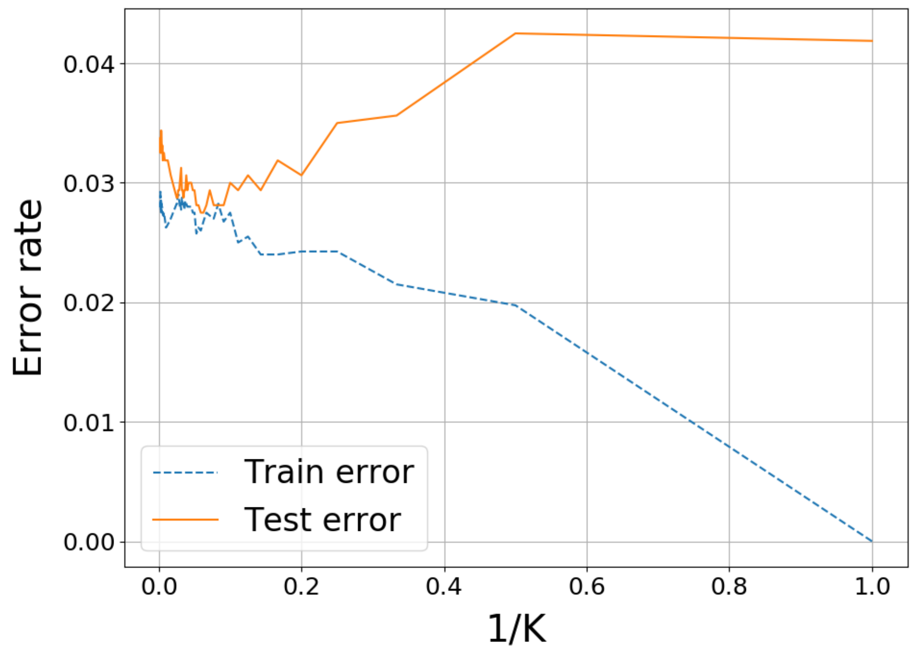


Figure 14: Training and test error of a K Nearest Neighbour Classifier as function of 1/K.

with a single validation set. Here, a subset of the training data is held out of the training process and the model evaluated on this validation set. In this case, the training only has to be done once, which reduces computation demands. Figure 15 shows an example of data set splitting in the case of $m = 3$.

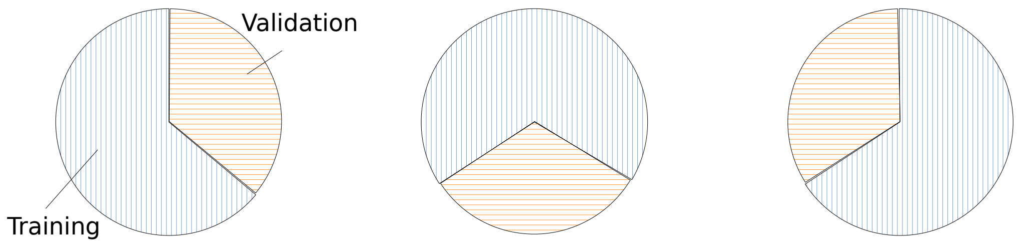


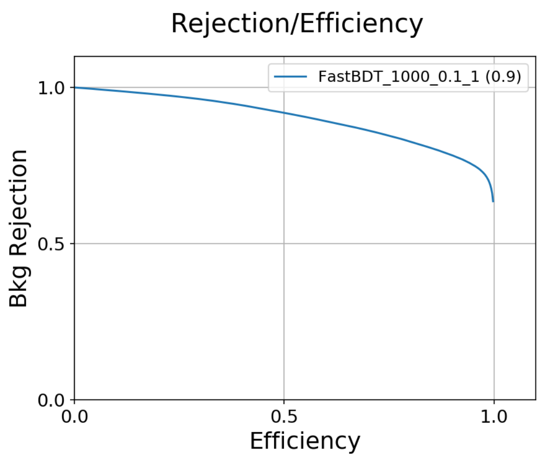
Figure 15: Exemplary splitting of a data set for three-fold cross-validation. The blue, vertically hatched part are used for training and the orange, horizontally hatched part is used for evaluation.

A way to evaluate the performance of a model in HEP applications is by plotting background rejection vs. signal efficiency and signal purity vs. signal efficiency curves and calculating the area under these curves. These quantities are defined as follows:

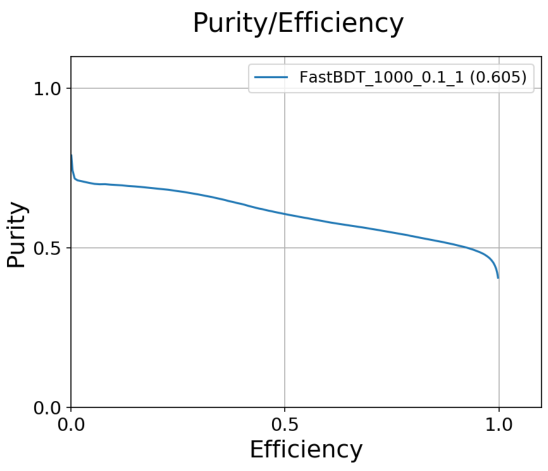
$$\begin{aligned}\text{Signal efficiency}: \ SE =& \frac{N_{sig,corr}}{N_{sig,tot}} \\ \text{Background rejection}: \ BR =& \frac{N_{bkg,corr}}{N_{bkg,tot}} \\ \text{Purity}: \ PU =& \frac{N_{sig,corr}}{N_{sig,corr} + N_{bkg,inc}}\end{aligned} \tag{26}$$

Here, $N_{sig,corr}, N_{bkg,corr}$ is the number of signal and background events respectively, which are correctly classified as such. $N_{bkg,inc}$ is the number of incorrectly classified background events. $N_{sig,tot}, N_{bkg,tot}$ are the total numbers of signal and background events respectively, meaning the correctly and incorrectly classified events. So signal efficiency is the proportion of correctly classified signal events. Background rejection is the proportion of correctly classified background events. Purity is the fraction of correctly identified signal events over the total number of identified signal events, which consists of correctly classified signal events and incorrectly classified background events. While signal efficiency and background rejection are fractions of signal events and background events respectively, they do not explicitly depend on the ratio of signal to background events. Purity does however and therefore one has to take into account the ratio of total signal to background events in the test set when evaluating purity.

Figure 16 shows examples of background rejection vs. signal efficiency and purity vs. signal efficiency plots.



(a) Background rejection vs. Signal efficiency



(b) Purity vs. Signal efficiency

Figure 16: Background rejection vs. Signal efficiency and Purity vs. Signal efficiency plots for a BDT classifier.

Both curves lie within the unit square. The area under the curve (AUC) for these plots is a measure of how well the model performs. The closer the curve approaches the top right corner of the unit square, the higher the AUC and the better the classification accuracy. With these measures, we can also compare the performance of different models.

4 Event Reconstruction

This section will describe the data sets which were used for the analysis and how this data was obtained.

The process of interest in this thesis is the purely muonic decay of a tau pair:

$$e^+e^- \to \tau^+\tau^- \to \mu^+(\nu_\tau\overline{\nu}_\mu)\ \mu^-(\overline{\nu}_\tau\nu_\mu) \tag{27}$$

This process is the signal for the following analysis and will be referred to as such in the subsequent sections. The final state consists of two muons and two neutrinos. Since the neutrinos cannot be detected, their presence can only be inferred by missing energy. Table 3 lists the signal process together with the processes which are considered to be the main sources of background and their cross-sections. Neutrinos are omitted.

Process	cross-section [nb]
$e^+e^- \to \tau^+\tau^- \to \mu^+\mu^-$	0.028
$e^+e^- \to \tau^+\tau^- \to 1prong$	0.271
$e^+e^- \to \mu^+\mu^-$	1.148
$e^+e^- \to e^+e^-\mu^+\mu^-$	18.9
$e^+e^- \to \mu^+\mu^-\mu^+\mu^-$	0.340
$e^+e^- \to e^+e^-$	300
$e^+e^- \to e^+e^-e^+e^-$	39.7
$e^+e^- \to q\overline{q}$	3.69
$e^+e^- \to mixed$	0.565
$e^+e^- \to charged$	0.535

Table 3: Signal and background processes together with the corresponding cross-sections.

The production of tau pairs at SuperKEKB has a cross-section of $0.919\,nb$. By multiplying this number with the branching ratio of a tau decaying to a muon and two neutrinos, we obtain the cross-section of the signal process:

$$\begin{aligned}\sigma(e^+e^- \to \tau^+\tau^- \to \mu^+\mu^-) &= \sigma(e^+e^- \to \tau^+\tau^-) \cdot BR(\tau^- \to \mu^-\nu_\tau\overline{\nu}_\mu)^2 \\ &= 0.919\,nb \cdot 0.1739^2 = 0.028\,nb\end{aligned} \tag{28}$$

We do the same for the decays $e^+e^- \to \tau^+\tau^- \to 1prong$. 1prong is a placeholder for three different final states containing a single charged particle:

- $e^+e^- \to \tau^+\tau^- \to e^+e^-$; $\mathrm{BR}(\tau^- \to e^-) = 0.1785$
- $e^+e^- \to \tau^+\tau^- \to \pi^+\pi^-$; $\mathrm{BR}(\tau^- \to \pi^-) = 0.1091$
- $e^+e^- \to \tau^+\tau^- \to \rho^+\rho^-$; $\mathrm{BR}(\tau^- \to \rho^-) = 0.2551$

$q\overline{q}$ denotes quark anti-quark pairs, mixed denotes decays to $B^0\overline{B^0}$ pairs and charged decays to B^+B^- pairs.

4.1 Pre-selection

For the event reconstruction, we first need to specify which pre-selection cuts should be applied. Here, we adhere to the selections made in [14] with a few changes due to the different event topology. The selection criteria are in detail:

Photons and π^0

The selection criteria for Photons and π^0s are identical to those in [14]. π^0s are reconstructed from Photon candidates, which satisfy:

- E > 0.1 GeV
- clusterNHits > 1.5
- -0.866 < $cos(\theta)$ < 0.9565 (CDC acceptance)
- 0.115 GeV < M_{π^0} < 0.152 GeV

The selection on M_{π^0} is applied to candidates for the process $\pi^0 \to \gamma\gamma$. Photons which are not used for the reconstruction of π^0, are subject to

- E > 0.2 GeV

Tracks

For the signal process, we expect two muons in the final state. We therefore select two charged tracks and imply a cut on the muon ID. Corresponding to the short lifetime of the tau, all its decay products are assumed to origin from a region around the interaction point. Furthermore, the two tracks shall lie within the barrel ECL region, i.e. the region of the detector which excludes the ECL endcaps. The applied selections are:

- muonID > 0.9
- -0.625 < $cos(\theta)$ < 0.846
- -3.0 < dz < 7.0 cm

- dr < 1.0 cm
- nGoodTracks $= 2$

We don't expect π^0s in the final state and therefore can select only tracks without those. Furthermore, we require less than three photons in the final state. This is a result of a preliminary analysis. These selections apply for both tracks:

- NPhotonsTrack1 < 3 , NPhotonsTrack2 < 3
- NPi0Track1 $= 0$, NPi0Track2 $= 0$

4.2 Data sets

We are dealing with three different data sets in total. One for the training of the machine learning models, one for evaluation of their performance and another one for the measurement of the cross-section of the process $e^+e^- \to \tau^+\tau^-$.

All three data sets are monte-carlo simulated (MC) data. The application of the models on real data requires careful validation and would have taken more time than feasible for this thesis. For all data sets, the same selection criteria apply.

Training data set

The data set for the training of the machine learning models was reconstructed from run-independent data. Run-independent data is not adjusted to specific conditions which vary from run to run, e.g. magnetic field settings or luminosity. We split the training data set into two parts, one for the actual training and one for validation. This validation data set will be used to get a first performance estimate of the models. There is no restriction on how large the training data set can be other than the available computer memory, so we want to use as many data points as possible. The validation data set however should resemble real data. The expected number of events per process can be calculated via the formula:

$$\hat{N_i} = \frac{\sigma_i}{\sigma_{signal}} * N_{signal} * \epsilon_{rec,i} \tag{29}$$

where σ_i is the cross-section of the i-th process, N_{signal} is the desired number of signal events in the data set and $\epsilon_{rec,i}$ is the efficiency of event reconstruction for the i-th process. Each process is scaled according to its cross-section and multiplied with the corresponding reconstruction efficiency. N_{signal} can be chosen arbitrarily, and we chose a number which achieves a compromise between data set size and purity.

Table 4 shows the number of events in the training data set, the scaling factor σ_i/σ_{signal}, the reconstruction efficiencies and the number of events in the validation data set.

Process	N_{train}	Scaling factor	ϵ_{rec}	N_{val}
$e^+e^- \to \tau^+\tau^- \to \mu^+\mu^-$	$2.21 \cdot 10^6$	1	0.395	2017
$e^+e^- \to \tau^+\tau^- \to 1prong$	$15 \cdot 10^3$	9.677	0.0001	4
$e^+e^- \to \mu^+\mu^-$	$4.51 \cdot 10^6$	41.071	0.013	2741
$e^+e^- \to e^+e^-\mu^+\mu^-$	$24 \cdot 10^6$	675	0.013	43800
$e^+e^- \to \mu^+\mu^-\mu^+\mu^-$	$112 \cdot 10^3$	12.143	0.068	4226
$e^+e^- \to e^+e^-$	0	10714	0	0
$e^+e^- \to e^+e^-e^+e^-$	14	1417	0	0
$e^+e^- \to q\bar{q}$	$1.27 \cdot 10^6$	131.786	0.002	1415
$e^+e^- \to mixed$	$583 \cdot 10^3$	20.178	0.003	354
$e^+e^- \to charged$	$1.06 \cdot 10^6$	19.107	0.007	643

Table 4: Number of training events, scaling factors, reconstruction efficiency and number of validation set events for each process.

All $e^+e^- \to e^+e^-$ and nearly all $e^+e^- \to e^+e^-e^+e^-$ events got rejected by the event reconstruction. The most import background processes are, as expected, those with at least two muons in the final state, namely $e^+e^- \to e^+e^-\mu^+\mu^-$, $e^+e^- \to \mu^+\mu^-$ and $e^+e^- \to \mu^+\mu^-\mu^+\mu^-$.

Test data set

In order to compare the trained methods and to make the necessary estimations, another, independent data set was gathered again using run-independent data. This data set and the data set on which the cross-section measurement is conducted are designed to be as close as possible to real data from the proc11 data taking campaign. proc11 is the combined luminosity-weighted data of the BELLE II experiments 7,8 and 10. The integrated luminosity of these experiments sums up to $L_{int} = 8764.2\,pb^{-1}$ for the $\Upsilon(4S)$ resonance. Additionally we assume a trigger efficiency of 90 %. The expected number of events for every process is calculated by:

$$N_{exp} = L_{int} \cdot \sigma \cdot 0.9 \tag{30}$$

Table 5 lists the expected event numbers and the number of actually reconstructed events for each process.

Measurement data set

Finally, to actually carry out the measurement, a third data set was constructed. Unaware to the author, the cross-section of the signal process was defined to be 17% smaller than the actual value of $0.919\,nb$:

$$\hat{\sigma}(e^+e^- \to \tau^+\tau^-) = 0.919\,nb \cdot 0.83 = 0.7628\,nb \tag{31}$$

Process	Expected events	Reconstructed events
$e^+e^- \to \tau^+\tau^- \to \mu^+\mu^-$	221 $\cdot 10^3$	87852
$e^+e^- \to \tau^+\tau^- \to 1prong$	2.138 $\cdot 10^6$	577
$e^+e^- \to \mu^+\mu^-$	9.055 $\cdot 10^6$	7839172
$e^+e^- \to e^+e^-\mu^+\mu^-$	149.079 $\cdot 10^6$	2967554
$e^+e^- \to \mu^+\mu^-\mu^+\mu^-$	2.682 $\cdot 10^6$	105019
$e^+e^- \to e^+e^-$	2366.334 $\cdot 10^6$	7
$e^+e^- \to e^+e^-e^+e^-$	313.145 $\cdot 10^6$	380
$e^+e^- \to q\overline{q}$	29.106 $\cdot 10^6$	69766
$e^+e^- \to mixed$	4.457 $\cdot 10^6$	35152
$e^+e^- \to charged$	4.220 $\cdot 10^6$	14077

Table 5: Expected number of events for proc11 data and actually reconstructed events for the test data.

This was a precautionary measure to uncover any bias which may have entered the analysis. This affects the processes with intermediate tau pair states leading to a 17 % decrease in the number of signal and $e^+e^- \to \tau^+\tau^- \to 1prong$ events. All other background events were created according to the expected numbers in table 5.

4.3 Variables

The variables used for the analysis are:

- Event variables:
  thrust, visibleEnergyOfEventCMS, missingMomentumOfEvent

- Muon variables:
  nPhotons, p, E, $cos(\theta)$, ϕ, E/p, E_{CMS}, p_{CMS}

- Tau variables:
  p, E, $cos(\theta)$, ϕ, E_{CMS}, p_{CMS}

Here, nPhotons is the number of photons detected in the region of the corresponding muon, p is the momentum and E the energy of the muon/tau. Variables sub-scripted with CMS are measured in the center-of-mass frame. Each muon and tau variable is measured for both muons and taus. The variables $cos(\theta)$ and ϕ refer to the azimuthal and polar angle measured in the coordinate system of the detector. MissingMomentumOfEvent refers to the momentum of the neutrinos and visibleEnergyOfEventCMS is the total energy visible energy, that is the energy of all final state particles except for neutrinos, in the center of mass frame. The thrust variable is the magnitude of the thrust vector. The thrust vector is defined as the unit vector in the direction of the axis along which the total projection of the momenta of all particles is maximized.

$$T = \frac{\sum |\vec{T} \cdot \vec{p_i}|}{\sum |\vec{p_i}|} \tag{32}$$

This particular choice of variables was made because of the relatively good performance of the machine learning models in a preliminary analysis.

A reduced variable set will be used for comparison with rectangular cuts:

- Reduced variable set:
  visibleEnergyOfEventCMS, missingMomentumOfEvent, thrust
  tau0_p_CMS, tau1_p_CMS, tau0_E_CMS, tau0_phi, tau1_phi and mu0_EoP

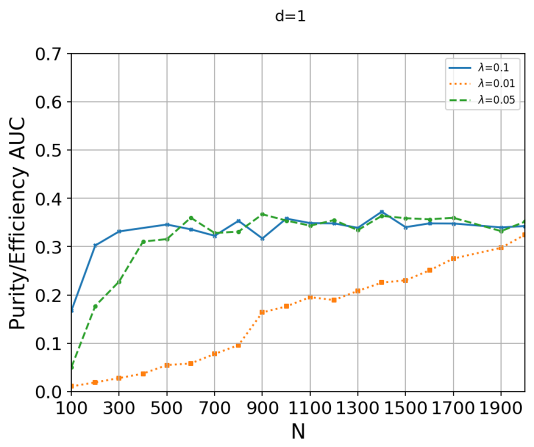
How this reduced set is chosen is described in detail in section 5.1.

5 Model selection

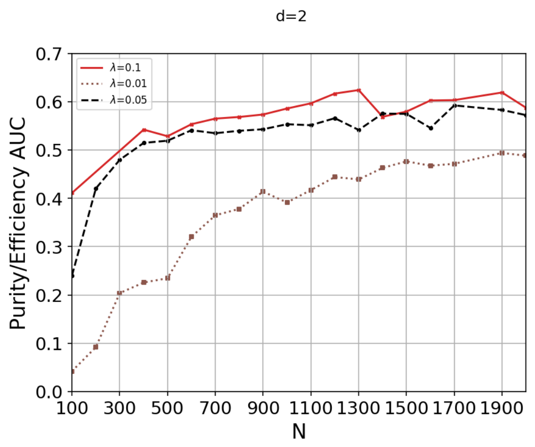
As a first step in the analysis, three different models were chosen, a BDT classifier, a GNB classifier and a MLP classifier. For the BDT, the FastBDT framework from BASF2 was chosen. The GNB classifier was implemented in Python with scikit-learn and the MLP is built with the Deep Learning frameworks Keras and TensorFlow. All three methods were used within the BASF2 framework to be able to compare them. Since the FastBDT is a BASF2 built-in method, we can simply use it with a function call. For the GNB and MLP however, it is necessary to overload some of the BASF2_MVA functions. The process of implementing and tuning the chosen methods shall be explained in more detail in the following sections.

5.1 FastBDT

As mentioned, FastBDT is the default machine learning method in BASF2. Its use is therefore straightforward. One only has to specify which variables and parameters to be used by it. To find out the optimal parameters for the training data set, we used a three-fold cross-validation approach. The training data set is split up randomly into two parts, a large one for training and a small one for evaluation. The metrics of evaluation are the AUC values of the background rejection vs. signal efficiency and of the purity vs. signal efficiency plots, in the following referred to as rejection and purity plots respectively. This procedure is repeated three times and the AUC values are then averaged. Figures 17 and 18 show the resulting AUC values plotted for different BDT parameters.

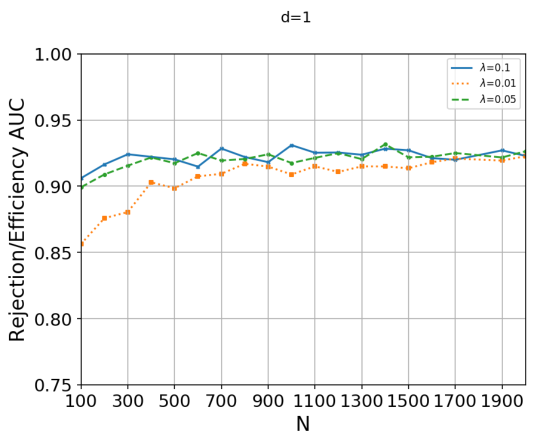


(a) d=1

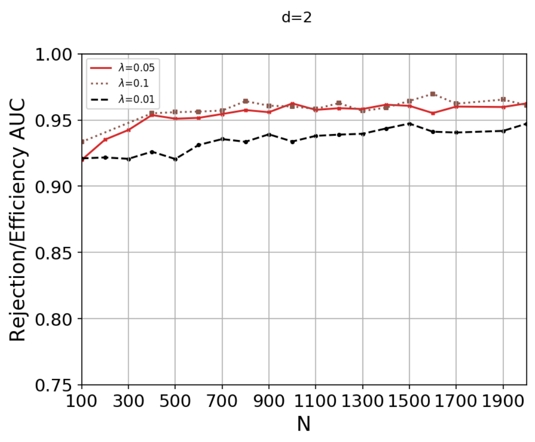


(b) d=2

Figure 17: Purity vs. signal efficiency AUC for different parameter values N, d, λ. On the left, results for d=1 are plotted, on the right for d=2.



(a) d=1



(b) d=2

Figure 18: Rejection AUC for different parameter values N, d, λ.

We used two different values d=1 and d=2 for the maximum tree depth, three different values $\lambda = 0.1, \lambda = 0.05$ and $\lambda = 0.01$ for the learning rates and 20 different values $N = 100 \ldots N = 2000$ for the number of trees. The AUC values of the purity plots increase for increasing N while the AUC values of the rejection plots stay nearly constant over the whole range of possible N values. Classifiers trained with $d = 2$ show overall a better performance than those trained with $d = 1$. The maximum values of purity plot AUC and rejection plot AUC are 0.602 and 0.970 respectively. The parameters with which these values were achieved are:

$$d = 2,\ \lambda = 0.1,\ N = 1600 \tag{33}$$

which were therefore used for further analysis.

The FastBDT is now trained with these parameters and the evaluated metrics are given in table 6.

Training time	2.32h
Rejection AUC	0.944
Purity AUC	0.516

Table 6: Training time and AUC value for rejection and purity plots of the FastBDT classifier.

The FastBDT has functions for the calculation of feature importance and correlations. Feature importance is the relative importance of each of the input variables, where importance in this sense means the distinctive power of the variable. In figure 19 the importance ranking of the input variables is plotted.

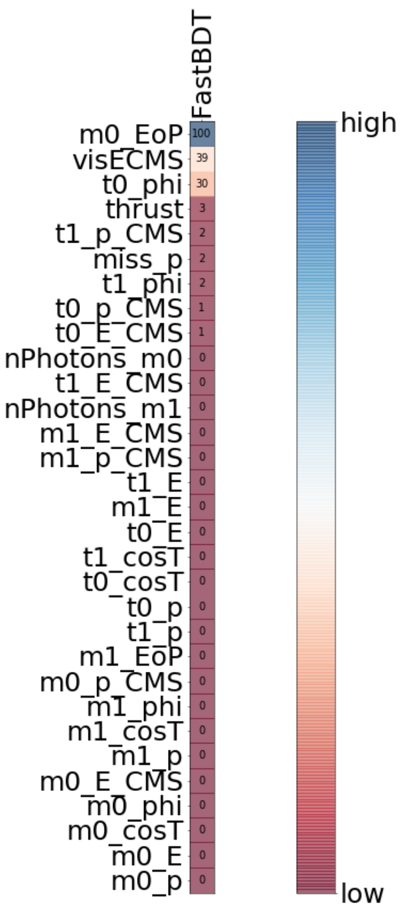


Figure 19: Importance ranking of the input variable set calculated from FastBDT.

The importance values are scaled to the interval $[0, 100]$ and abbreviations for the variable names introduced. t0, t1, mu0, mu1 label the first and the second tau and muon respectively. As can be seen, the importance values drop rapidly after even the first variable. After the ninth variable, all importance values are zero. We use this fact to construct a reduced variable set, which consists only of the top nine variables: visibleEnergyOfEventCMS, missingMomentumOfEvent, thrust tau0__p__CMS, tau1__p__CMS, tau0__E__CMS, tau0__phi, tau1__phi and mu0__EoP . The idea is that variables with little to zero importance will not contribute to the overall performance of the classifier and can therefore be left out.

With this reduced variable set, we can train and evaluate the FastBDT again. Results are shown in table 7.

Figure 20 shows the rejection and purity plots for both the full and the reduced variable set. The performance of the classifier in terms of the rejection AUC is nearly identical.

Training time	1h 12min
Rejection AUC	0.943
Purity AUC	0.491

Table 7: Metrics of the FastBDT classifier trained on the reduced variable set.

The purity AUC value decreases by 5% going from the full to the reduced variable set. The performance of the classifier trained on the reduced variable set is comparable to the one trained on the full variable set while the training time nearly halved.

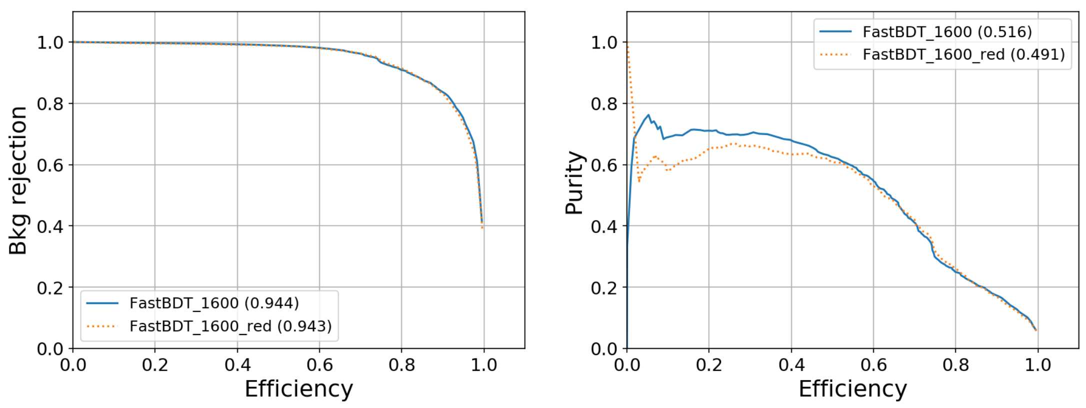


Figure 20: Rejection and purity plots of FastBDT for full and reduced variables. The dotted line shows the plots for reduced variables. In braces are the AUC values for the corresponding classifier.

Figure 21 shows the correlations of the input variables in the reduced variable set for both the signal and background. Positive correlations are indicated by blue coloring and positive values and negative correlations by red coloring and negative values. As can be seen, correlations are lower for signal than for background events. Another way of reducing the number of input variables is by dismissing those variables which are highly correlated with others, until the correlations fall below a threshold value. The idea here is that pairs of highly correlated variables do not provide more information than any one of these variables alone. We however, don't modify dismiss any further variables and accept some correlations between them.

5.2 Multilayer Perceptron

For the MLP classifier, we use the Deep Learning framework Keras. Keras is a high-level API which can work in conjunction with the Deep Learning platform TensorFlow. In order to build an MLP classifier with Keras and BASF2, we need to rewrite some of the functions from the BASF2_MVA module e.g. the fit function. As described in section 3.5, we define the the topology of the MLP to be as easy as possible, one hidden layer

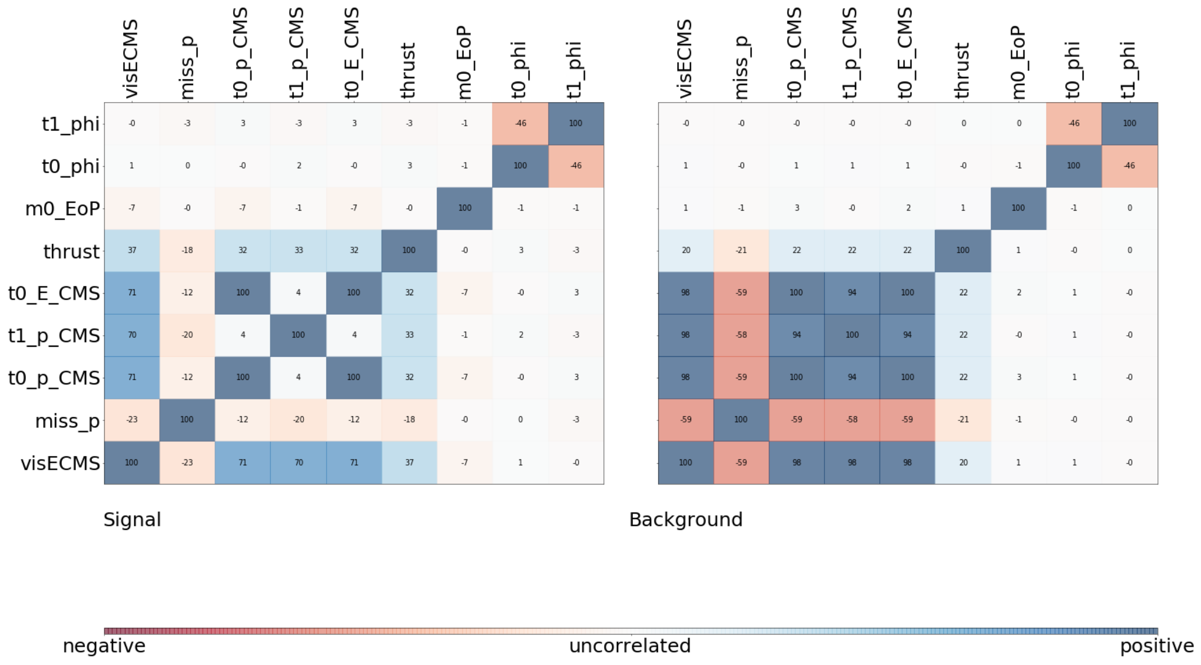


Figure 21: Correlation matrix for the reduced input variable set calculated from FastBDT.

with a number of neurons equal to $(N_{variables} + 1)/2$. We choose a sigmoid function as the activation function and binary cross-entropy as the loss function. The properties of the MLP are given in table 8.

Input Neurons	$N_{variables}$
Hidden Neurons	$\frac{(N_{variables+1})}{2}$
Output Neurons	1
activation function	sigmoid
loss function	binary cross-entropy
optimization algorithm	ADAM

Table 8: Properties of the MLP classifier.

The MLP does not have hyper-parameters which can be optimized by cross-validation. During the training phase, the trained model is evaluated on a small subset of the training data in every iteration. If the difference in performance per iteration falls below a certain threshold, the training is stopped. This evaluation data set can be customized to resemble expected future data. However, comparison with a random chosen evaluation data set did not show significant differences in performance on unseen data. The MLP classifier cannot determine feature importance values or feature correlations. Table 9 shows the evaluated metrics of the MLP classifier for the full and the reduced variable set.

	full variables	reduced variables
Training time	1h 10min	41min
Rejection AUC	0.968	0.951
Purity AUC	0.692	0.547

Table 9: Metrics of the MLP classifier trained on the full and reduced variable set.

For the MLP, the difference in training times for full and reduced variable sets are not as high as for the FastBDT. Figure 22 shows the Rejection and purity plots for both the full and the reduced variable set. The performance is again worse for reduced variables with a drop in the AUC value for rejection by 1% and a decrease by 26% for purity. This corresponds to a difference of 0.017 and 0.145 in absolute values for rejection and purity respectively.

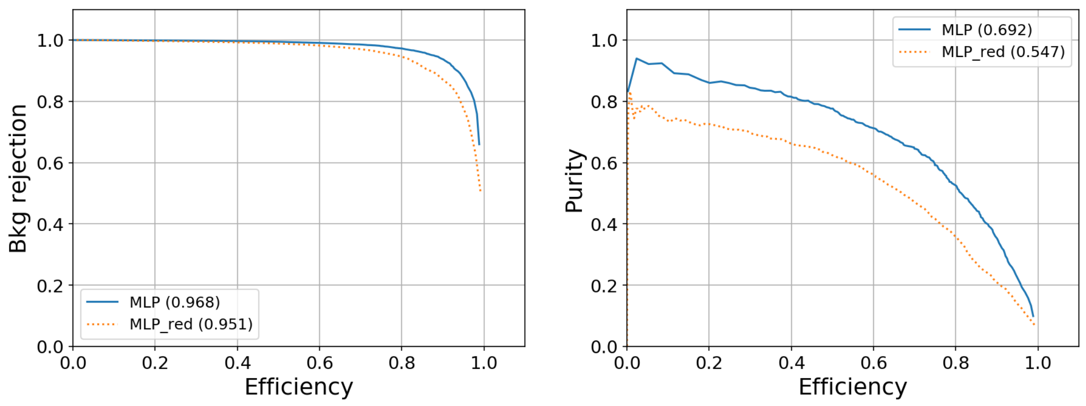


Figure 22: Rejection and purity plots of MLP for full and reduced variables. The dotted line shows the plots for reduced variables. In braces are the AUC values.

5.3 Gaussian Naive Bayes

The GNB classifier is part of the scikit-learn Python library. Its implementation is similar to the one of the MLP, however there are no options to specify. The conditional probability $P(x_i|y)$ is defined by the method itself. This makes the GNB classifier easier to use but at the same time more rigid. There is no way to adapt the classifier to the characteristics of the problem, which can lead to worse performance. As the MLP, it does not have the possibility of determining the feature importance and feature correlations.

Table 10 shows the metrics of the GNB classifier evaluated on the validation set for the full and reduced variables.

Figure 23 shows the rejection and purity plots for both the full and the reduced variable set. The classifier trained on the reduced set apparently performs better in this case.

	full variables	reduced variables
Training time	4min 46sec	1min 42sec
Rejection AUC	0.878	0.915
Purity AUC	0.126	0.376

Table 10: Metrics of the GNB classifier trained on the full and reduced variable set.

The comparison is not fully conclusive however, because not the full range of signal efficiencies is covered by the GNB trained on the full variable set. The AUC values for the GNB trained on the reduced variable set increase by 4% or 0.037 for rejection and by a factor of roughly 3 for purity. With training times of only about 5 and 2 minutes for full and reduced variables respectively, the GNB classifier is significantly faster in training than the other methods for this data set.

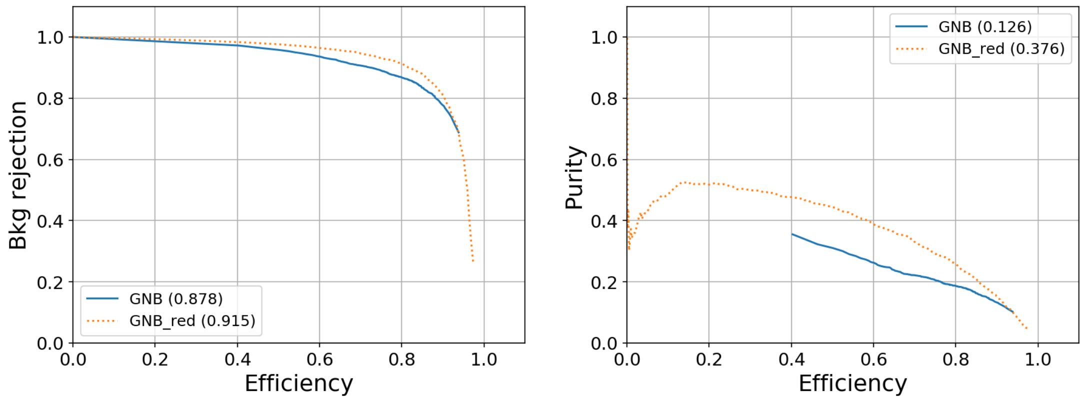


Figure 23: Rejection and purity plots of GNB for full and reduced variables. The dotted line shows the plots for reduced variables. In braces are the AUC values.

6 Analysis

This section is concerned with the selection of rectangular cuts, the comparison with the machine learning models and with a possible combination of the two methods.

6.1 Rectangular cuts

The reason we chose to construct a reduced variable set is to keep the analysis with rectangular cuts simpler. As shown in section 5, the performance of the machine learning models is comparable for both variable sets, but to choose cuts on the 31 variables in the full variable set is significantly more complex. Due to the correlations of the variables, many of them will have no marginal effects and would therefore be dismissed anyway, so we restrict ourselves to the reduced variable set. Additionally, the angular variables tau0_phi and tau1_phi were excluded for the analysis with rectangular cuts. The reason for this is that these variables are nearly uniformly distributed and there is no reasonable way of applying cuts on them. This issue is displayed in figure 24.

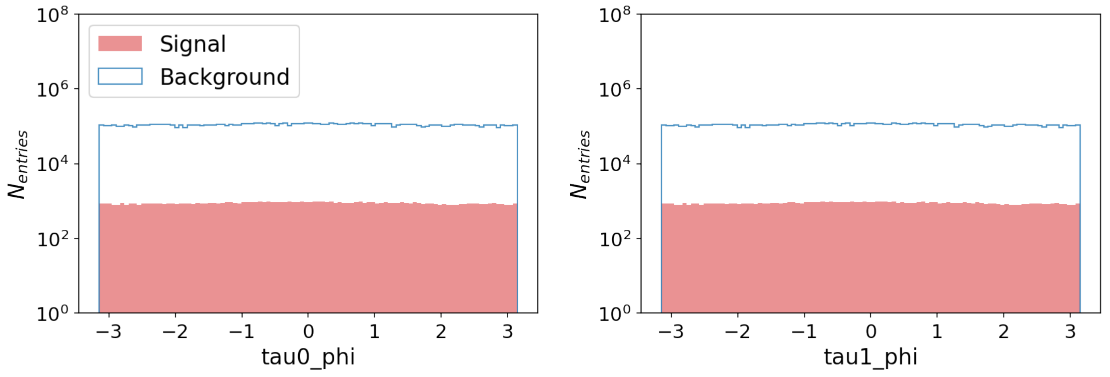


Figure 24: Distributions of angular variables tau0_phi and tau1_phi for signal (red, shaded) and background (blue, unshaded) in the test data set.

We make the assumption here, that even when we forego using the phi variables, the performance of the manual cut analysis will not be impaired by much. This leads to the following variables for cut-based analysis:

- tau0_p_CMS, tau1_p_CMS, tau0_E_CMS
- mu0_EoP
- visibleEnergyOfEventCMS,
- missingMomentumOfEvent,
- thrust

To choose cuts, we first plot the distributions of the variables for signal and background in the test data set. Cuts are applied by visually selecting regions which should be excluded from the data. Figure 25 shows the distributions of the relevant variables. The vertical lines indicate the value of the variable at which it was cut.

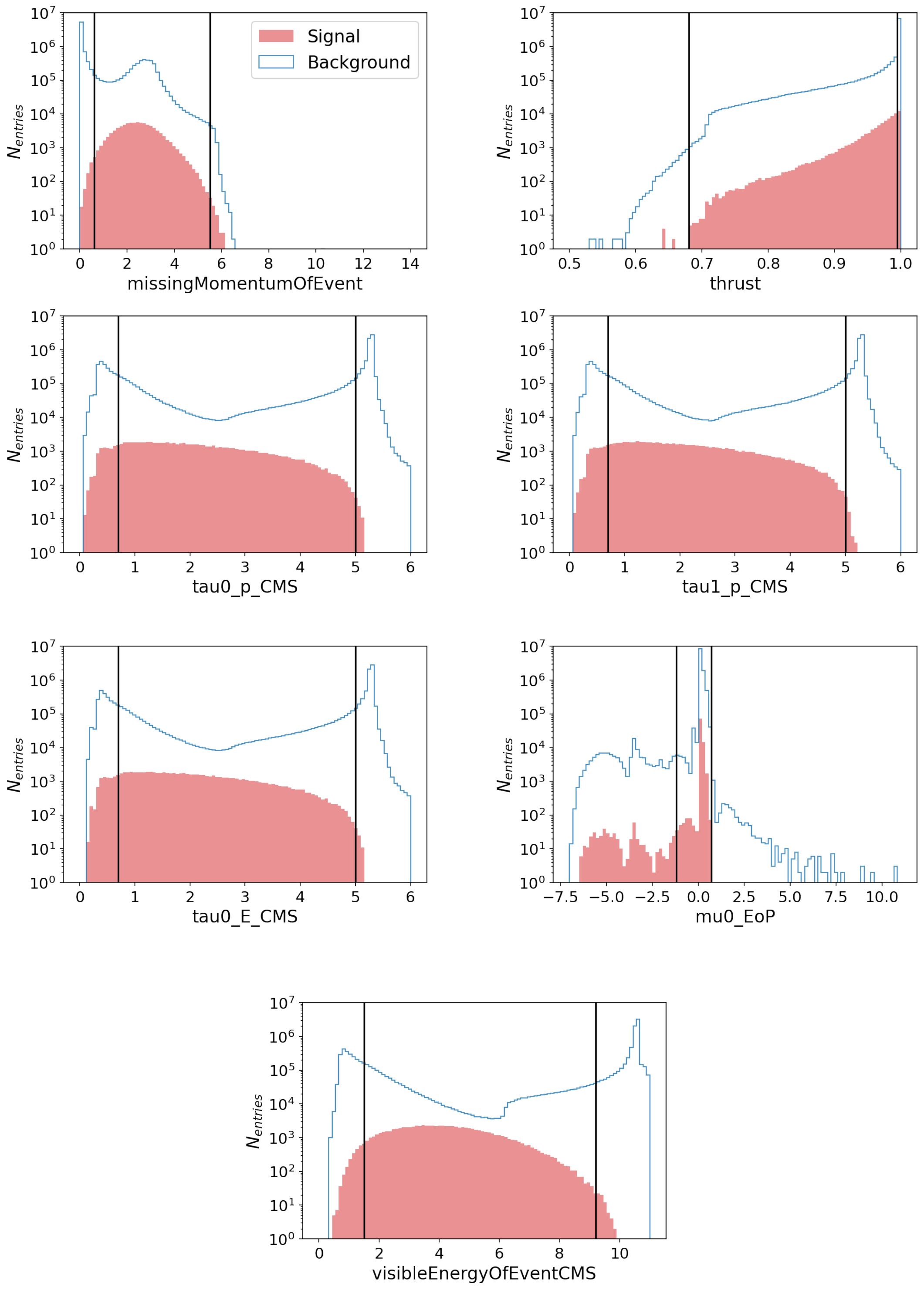


Figure 25: Distributions of variables missingMomentumOfEvent, thrust, tau0__p__CMS, tau1__p__CMS, tau0__E__CMS, mu0__EoP and visibleEnergyOfEventCMS for signal (red, shaded) and background (blue, unshaded) in the test data set. The vertical black lines indicate the cuts.

After an analysis of the marginal effects, the variable tau0_E_CMS was excluded. This is because it is highly correlated with the tau0_p_CMS variable and placing a cut on one of these variables has nearly identical effects on the other. Finally, the applied cuts are:

- 0.7 < tau0_p_CMS, tau1_p_CMS < 5
- -1.2 < mu0_EoP < 0.7
- 0.6 < missingMomentumOfEvent < 5.5
- 0.68 < thrust < 0.995
- 1.5 < visibleEnergyOfEventCMS < 9.2

These cut-values lead to a signal efficiency of 0.672, a background rejection of 0.877 and a purity of 0.041.

6.2 Comparison

The results of these cuts and the comparison with the classifiers are listed in table 11. The values of the classifiers are shown for the signal efficiency 0.672.

Method	Background rejection	Purity
Rectangular cuts	0.877	0.041
MLP	0.987	0.289
FastBDT	0.985	0.258
GNB	0.979	0.200

Table 11: Purity and background rejection for the different methods at signal efficiency 0.672.

Figure 26 shows the rejection and purity plots for all three classifiers together with the calculated values for manual cutting.

Of the three classifiers, the MLP performs the best, the FastBDT and GNB are similar in performance with the GNB achieving lower AUC values for the purity curve. With manual cuts, the values for background rejection and purity achieved are considerably lower for fixed efficiency than those of all three classifiers.

As a comparison, figure 27 shows the same plots for all three classifiers trained on the full variable set.

The ranking is the same as with the reduced variables. Since the MLP classifier has the best performance on the test data set, we will use it for the cross-section measurement. Also, we will use the full variable set for the measurement, to achieve the best possible results. But what values of purity and background rejection should be used? To answer

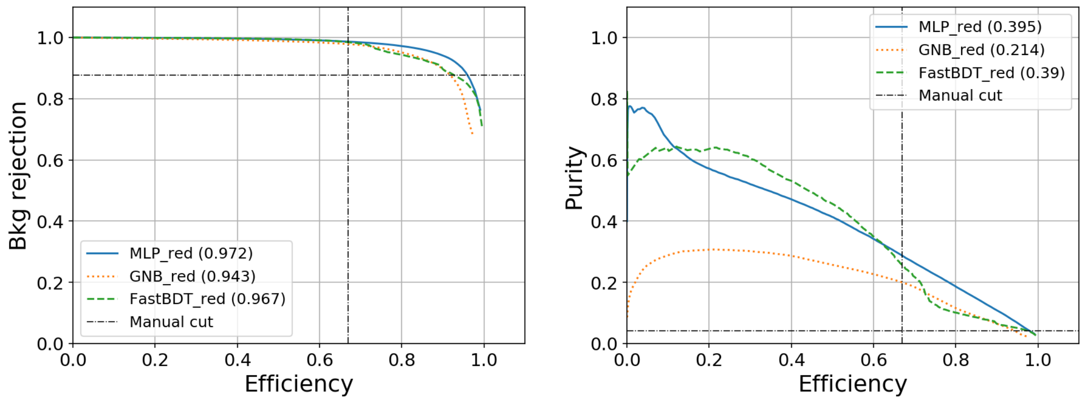


Figure 26: Rejection and purity plots for FastBDT, MLP, GNB and rectangular cuts. The dash-dotted lines indicate the achieved values for the rectangular cuts.

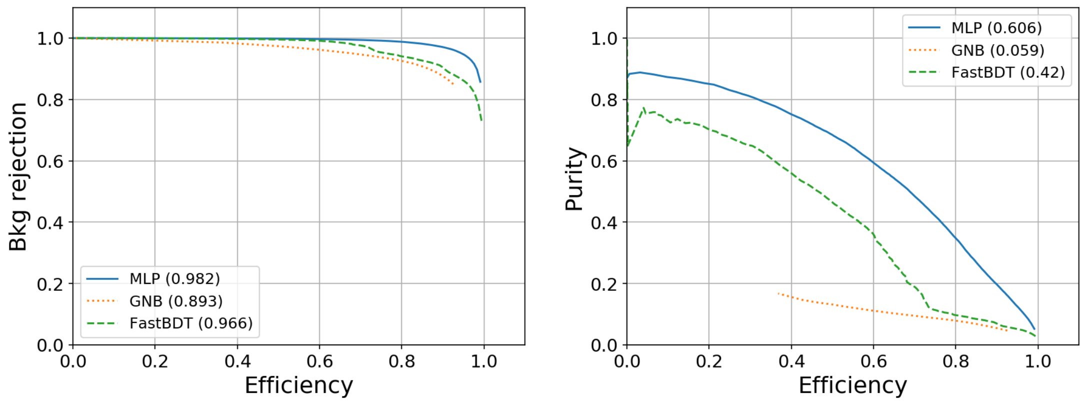


Figure 27: Rejection and purity plots for FastBDT, MLP and GNB trained on full variables.

this question, we first look at how the classification is done in practice. After applying the MLP to the test data set, it assigns a real value in the interval (0, 1) to each data point. Values closer to 0 indicate a higher change of being a background event and vice versa for signal. We interpret this number as the probability of being a signal event. The classification is then simply a cut on this signal probability. We can plot the metrics as functions of the signal probability. This is done in figure 28.

Both the background rejection and the purity are increasing functions while the signal efficiency is a decreasing function of signal probability. One method of choosing a cut is to take the signal probability value at the intersection of the curves of purity and signal efficiency, if it exists. A more systematic approach is to maximize a figure of merit (FOM). We choose to maximize the Punzi FOM, which is defined by:

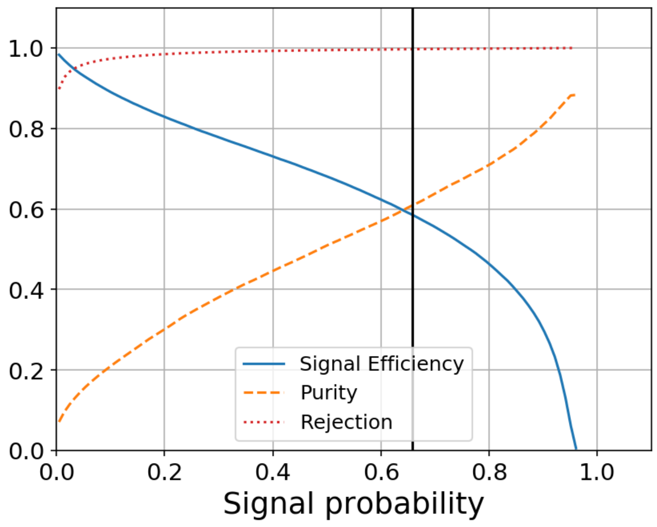


Figure 28: Background rejection, signal efficiency and purity of the MLP trained on the full variable set as functions of the signal probability. The vertical black line indicates the chosen cut at 0.658.

$$Punzi\ FOM = \frac{N_{sig}}{\sqrt{N_{sig} + N_{bkg}}} \tag{34}$$

The maximum value of the FOM is 177.07 and is achieved when cutting at signal probability 0.658. This cut leads to a signal efficiency of 0.589, a purity of 0.606 and a background rejection of 0.996. Figure 29 shows the histogram of signal probabilities for the test data set together with the chosen cut.

All events with signal probability lower than 0.658 will be treated as background events and those with higher signal probabilities as signal events.

6.3 Combination of MLP with rectangular cuts

We now have established which machine learning method we will be using for the measurement and at which signal probability the cut should be placed. Before actually doing the measurement, we take a look at the variable distributions after the MLP is applied. Figure 30 shows the distributions for the thrust variable before and after the MLP classifier.

We can place a cut on the distribution of the thrust variable after the MLP was applied and can get rid of further background events. When no signal events are being cut, the signal efficiency remains unchanged.

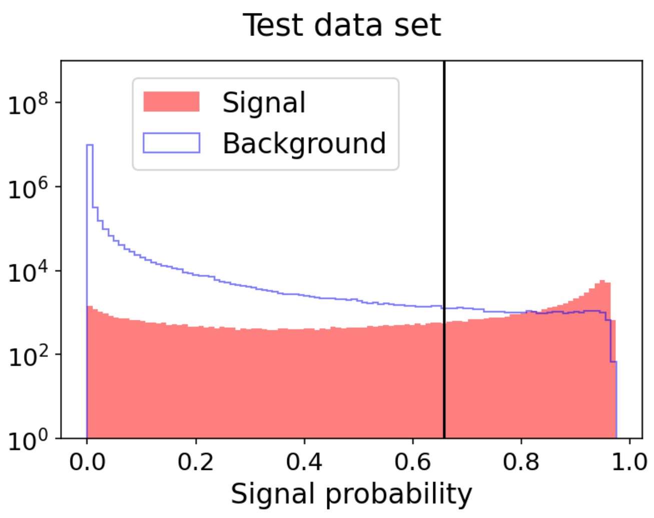


Figure 29: Histogram of the MLP classifier output for signal (red, shaded) and background (blue, unshaded) events in the test data set. The vertical black line indicates the cut at 0.658.

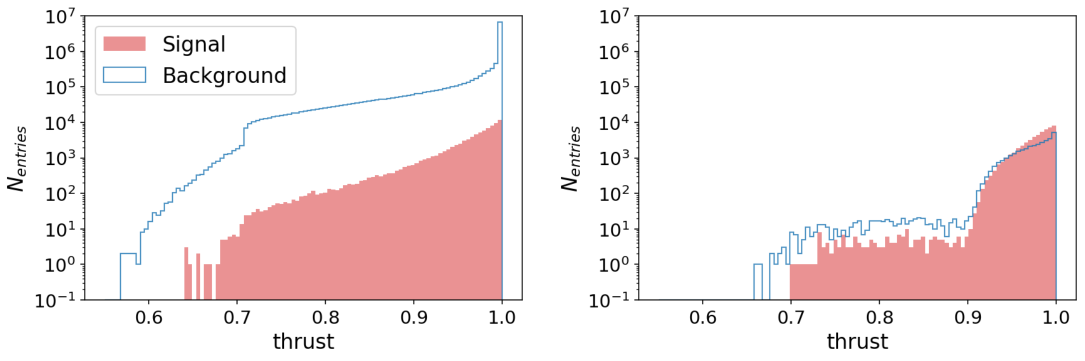


Figure 30: Distributions of the thrust variable for signal (red, shaded) and background (blue, unshaded) before (left) and after (right) application of the MLP classifier. MLP cut applied at 0.658 signal probability.

Figure 31 shows the distributions of all variables in the reduced variable set after application of the MLP at signal probability 0.658. The black lines indicate the cuts on the variables. The cuts were chosen so that no additional signal event was lost. Marginal effects were neglected in this case.

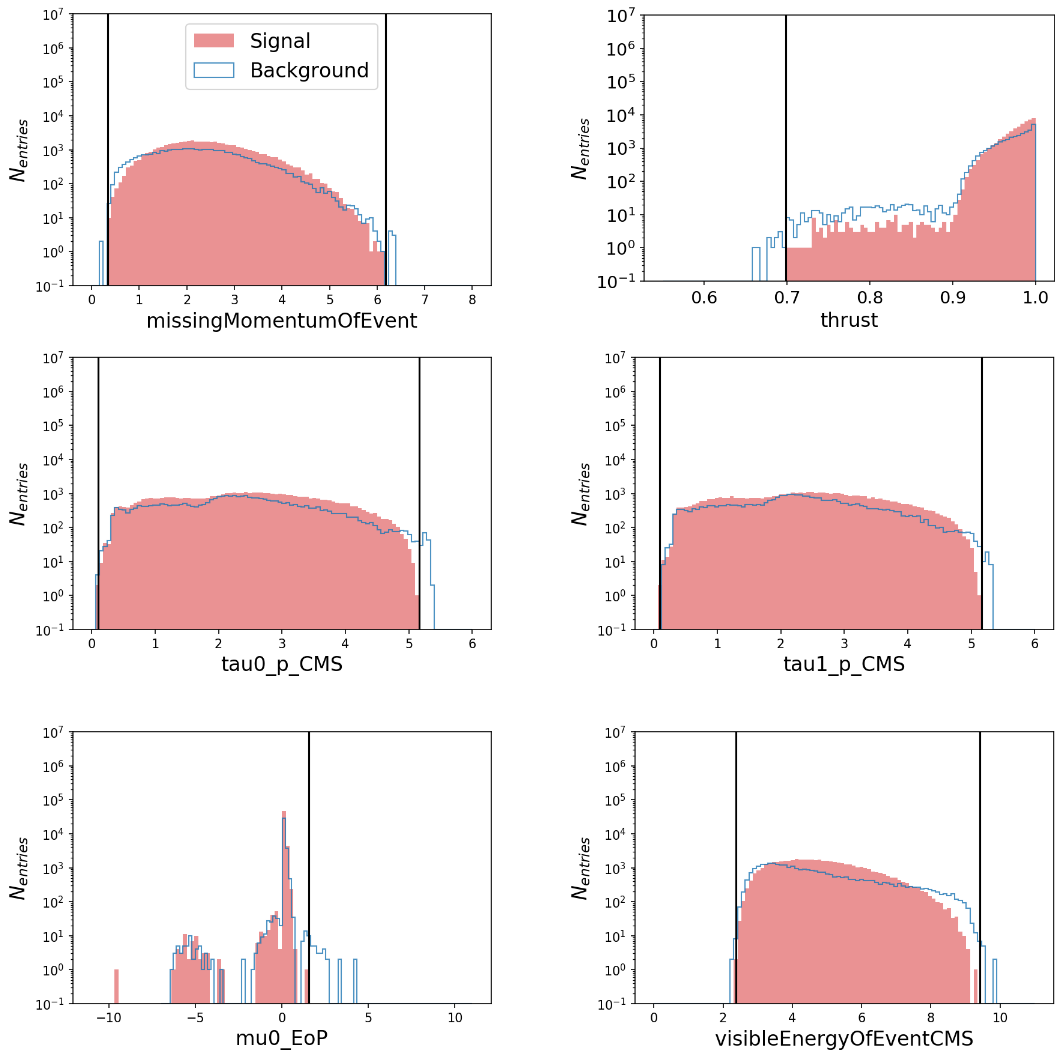


Figure 31: Distributions of variables missingMomentumOfEvent, thrust, tau0__p__CMS, tau1__p__CMS and visibleEnergyOfEventCMS for signal (red, shaded) and background (blue, unshaded) in the test data set after application of the MLP classifier. The vertical black lines indicate the cuts.

The full list of applied cuts is:

- $0.658 <$ Signal probability
- $0.70 <$ thrust
- $2.37 <$ visibleEnergyOfEventCMS < 9.32
- $0.34 <$ missingMomentumOfEvent < 6.08
- $0.10 <$ tau0_p_CMS < 5.11
- $0.09 <$ tau1_p_CMS < 5.11
- mu0_EoP < 1.34

With these cuts, an additional 351 background events could be dismissed. The performance gain however is minimal in this case and did not change the purity, signal efficiency or background rejection for the given precision. For the given problem, the combination of machine learning model and rectangular cuts does not lead to improved performance. Moreover, as will be seen in the following chapter, the measurement of the cross-section is less accurate when combining the two methods. This is an indicator, that it the combination of rectangular cuts with machine learning models is not straightforward. However, further investigations could be made in this subject.

7 Results

This section is now concerned with the measurement of the cross-section of the process $\sigma(e^+e^- \to \tau^+\tau^-)$. In order to obtain the cross-section, we need to measure the number of signal events in the decay $e^+e^- \to \tau^+\tau^- \to \mu^+\mu^-$. We expect the measurement data set to contain a number of signal events equal to

$$N_{sig} = \sigma(e^+e^- \to \tau^+\tau^-) \cdot BR(\tau^- \to \mu^-\nu_\tau\overline{\nu}_\mu)^2 \cdot L_{int} \cdot \epsilon_{rec} \cdot \epsilon_{trg} \cdot \epsilon_{sig} \tag{35}$$

Here, $BR(\tau^- \to \mu^-\nu_\tau\overline{\nu}_\mu)$ is the branching ratio of a tau decaying into a muon. This branching ratio is squared since we need to consider both τ^- and τ^+ decaying into muons, which have identical branching ratios. L_{int} is the integrated luminosity, ϵ_{trg} is the trigger efficiency, ϵ_{rec} is the reconstruction efficiency of the signal process and ϵ_{sig} the signal efficiency of the chosen method of analysis. We reshape the equation to

$$\sigma(e^+e^- \to \tau^+\tau^-) = \frac{N_{sig}}{BR(\tau^- \to \mu^-\nu_\tau\overline{\nu}_\mu)^2 \cdot L_{int} \cdot \epsilon_{rec} \cdot \epsilon_{trg} \cdot \epsilon_{sig}} \tag{36}$$

thus obtaining a formula for the cross-section. $BR(\tau^- \to \mu^-\nu_\tau\overline{\nu}_\mu)$ and L_{int} are known and ϵ_{trg} is assumed to be 90%. N_{sig} can be obtained via the MLP or by applying rectangular cuts. Since it is not known if an event from the measurement data set is a signal or a background event, we have to estimate N_{sig} with the formula

$$\begin{aligned} N_{sig} &= N_{sig} + N_{bkg} \cdot \frac{N_{sig}}{N_{sig} + N_{bkg}} = N_{cand} \cdot purity \\ N_{cand} &= N_{sig} + N_{bkg} \end{aligned} \tag{37}$$

where N_{cand} is the number of correctly classified signal events and the number of incorrectly classified background events. We multiply the number of signal candidates N_{cand} by the purity as it was defined above. Since the purity is calculated from the test data set and unknown in the measurement data set, we can only get an estimation of the number of signal events. The reconstruction efficiency ϵ_{rec} and the signal efficiency ϵ_{sig} are also calculated from the test data set. Thus we arrive at the formula

$$\sigma(e^+e^- \to \tau^+\tau^-) = \frac{N_{cand} \cdot purity}{BR(\tau^- \to \mu^-\nu_\tau\overline{\nu}_\mu)^2 \cdot L_{int} \cdot \epsilon_{rec} \cdot \epsilon_{trg} \cdot \epsilon_{sig}} \tag{38}$$

The uncertainty in the branching ratio is known and we assume zero uncertainty for L_{int} and ϵ_{trg}. For the calculation of the other uncertainties, we assume independence of the respective quantities. Since the measurement of a number of N_{cand} events can be interpreted as a counting experiment, we can assume that number of candidate events is following a poisson distribution with mean N_{cand}. The uncertainty in N_{cand} is therefore

$$\delta(N_{cand}) = \sqrt{N_{cand}} \tag{39}$$

The same assumptions hold for the number of signal and background events respectively:

$$\begin{aligned} \delta(N_{sig}) &= \sqrt{N_{sig}} \\ \delta(N_{bkg}) &= \sqrt{N_{bkg}} \end{aligned} \tag{40}$$

The relative uncertainty in the number of candidate events is therefore:

$$f_{N_{cand}} = \frac{N_{cand}}{\delta(N_{cand})} = \frac{1}{\sqrt{N_{cand}}} \tag{41}$$

For the calculation of the uncertainty in the purity, we make use of propagation of uncertainty:

$$\begin{aligned} f^2_{purity} = \left(\frac{\delta(purity)}{purity}\right)^2 &= \left(\frac{\delta(N_{sig})}{N_{sig}}\right)^2 + \left(\frac{\delta(N_{sig}+N_{bkg})}{N_{sig}+N_{bkg}}\right)^2 \\ &= \left(\frac{1}{\sqrt{N_{sig}}}\right)^2 + \frac{\delta(N_{sig})^2 + \delta(N_{bkg})^2}{(N_{sig}+N_{bkg})^2} \\ &= \frac{1}{N_{sig}} + \frac{N_{sig}+N_{bkg}}{(N_{sig}+N_{bkg})^2} \\ &= \frac{1}{N_{sig}} + \frac{1}{N_{sig}+N_{bkg}} \end{aligned} \tag{42}$$

We obtain the relative uncertainty in purity:

$$f_{purity} = \sqrt{\frac{1}{N_{sig}} + \frac{1}{N_{sig}+N_{bkg}}} \tag{43}$$

The relative uncertainties in ϵ_{rec} and ϵ_{sig} are calculated in the same manner:

$$
\begin{aligned}
f^2_{\epsilon_{rec}} &= \left(\frac{\delta\epsilon_{rec}}{\epsilon_{rec}}\right)^2 = \left(\frac{\delta N_{rec}}{N_{rec}}\right)^2 = \frac{1}{N_{rec}} \rightarrow f_{\epsilon_{rec}} = \sqrt{\frac{1}{N_{rec}}} \\
f^2_{\epsilon_{sig}} &= \left(\frac{\delta\epsilon_{sig}}{\epsilon_{sig}}\right)^2 = \left(\frac{\delta N_{sig}}{N_{sig}}\right)^2 = \frac{1}{N_{sig}} \rightarrow f_{\epsilon_{sig}} = \sqrt{\frac{1}{N_{sig}}}
\end{aligned} \tag{44}
$$

Here, N_{rec} is the number of reconstructed events and N_{sig} is the number of correctly identified signal events in the test data set. Note that ϵ_{rec}, ϵ_{trg}, L_{int} and $BR(\tau^- \rightarrow \mu^-\nu_\tau\overline{\nu}_\mu)$ are independent of the method of analysis. Their values are:

$$
\begin{aligned}
\epsilon_{rec} &= 0.4009 \pm 0.0014\ ;\ f_{\epsilon_{rec}} = 0.34\% \\
BR(\tau^- \rightarrow \mu^-\nu_\tau\overline{\nu}_\mu) &= 0.1739 \pm 0.0004\ ;\ f_{BR} = 0.23\% \\
\epsilon_{trg} &= 0.9 \\
L_{int} &= 8764200\, nb^{-1}
\end{aligned} \tag{45}
$$

MLP

The quantities obtained with the analysis with the MLP classifier are:

$$
\begin{aligned}
N_{cand} &= 71145 \pm 266.73\ ;\ f_{N_{cand}} = 0.37\% \\
purity &= 0.6059 \pm 0.0034\ ;\ f_{purity} = 0.56\% \\
\epsilon_{sig} &= 0.5888 \pm 0.0026\ ;\ f_{\epsilon_{sig}} = 0.44\%
\end{aligned} \tag{46}
$$

The calculated cross-section is:

$$
\sigma(e^+e^- \rightarrow \tau^+\tau^-) = \frac{71145 \cdot 0.6059}{0.1739^2 \cdot 8764200 nb^{-1} \cdot 0.4009 \cdot 0.9 \cdot 0.5888} = 0.7656\, nb \tag{47}
$$

To obtain the relative uncertainty in the cross-section, we sum up the squared relative uncertainties:

$$
\begin{aligned}
f_\sigma &= \sqrt{f^2_{N_{cand}} + f^2_{purity} + f^2_{BR} + f^2_{\epsilon_{rec}} + f^2_{\epsilon_{sig}}} \\
&= \sqrt{0.0037^2 + 0.0056^2 + 0.0023^2 + 0.0034^2 + 0.0044^2} \\
&= 0.90\%
\end{aligned} \tag{48}
$$

The final result is:

$$\sigma(e^+e^- \to \tau^+\tau^-) = 0.7656 \pm 0.0069\, nb \tag{49}$$

Rectangular cuts

For the analysis with the rectangular cuts, we obtain:

$$\begin{aligned} N_{cand} &= 1158313 \pm 1076.25 \ ; \ f_{N_{cand}} = 0.09\% \\ purity &= 0.0415 \pm 0.0002 \ ; \ f_{purity} = 0.42\% \\ \epsilon_{sig} &= 0.6696 \pm 0.0028 \ ; \ f_{\epsilon_{sig}} = 0.41\% \end{aligned} \tag{50}$$

which leads to the cross-section:

$$\begin{aligned} &\sigma(e^+e^- \to \tau^+\tau^-) = 0.7507 \pm 0.0054\, nb \\ &f_\sigma = 0.72\% \end{aligned} \tag{51}$$

Combination

When combining the two methods, we obtain:

$$\begin{aligned} N_{cand} &= 70203 \pm 264.96 \ ; \ f_{N_{cand}} = 0.37\% \\ purity &= 0.606 \pm 0.0033 \ ; \ f_{purity} = 0.56\% \\ \epsilon_{sig} &= 0.5888 \pm 0.0026 \ ; \ f_{\epsilon_{sig}} = 0.44\% \end{aligned} \tag{52}$$

$$\begin{aligned} &\sigma(e^+e^- \to \tau^+\tau^-) = 0.7556 \pm 0.0068\, nb \\ &f_\sigma = 0.90\% \end{aligned} \tag{53}$$

As stated above, the cross-section with which the data was generated is 0.7628 nb. Figure 32 shows a comparison of the measurements. The true value lies within the uncertainty calculated with the MLP. It is not within the uncertainty calculated via rectangular cuts or via a combination of the two methods. Possible explanations are, that the uncertainty was underestimated due to the neglect of uncertainty in the integrated luminosity or that systematic errors arise in the procedure. The combination of both methods does lead to a significant improvement over the case where only rectangular cuts are used.

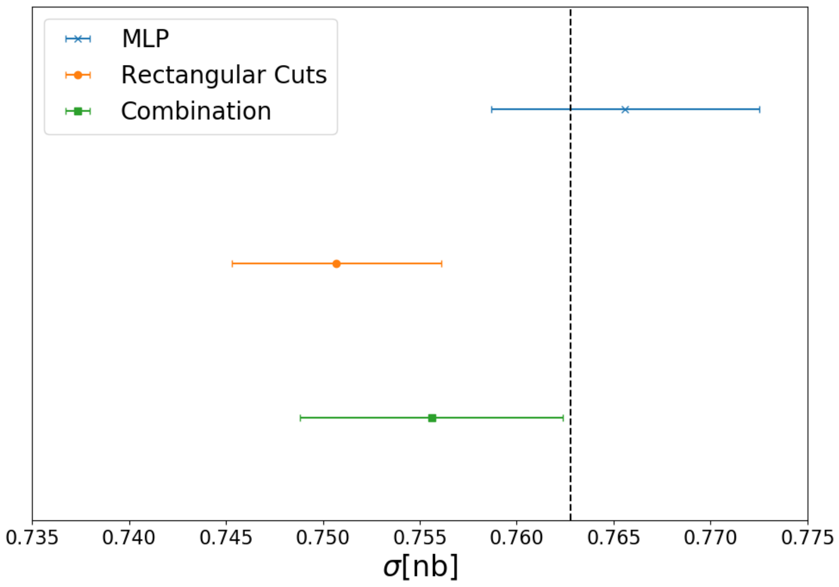


Figure 32: Comparison of the measured cross-section for MLP, rectangular cuts and a combination of the two. The vertical black line indicates the true value of the cross-section.

8 Conclusion and Outlook

To answer the main hypothesis of this thesis: machine learning models can classify muonic decays of tau-pairs better in terms of signal efficiency, background rejection and purity than what is possible with simple cut-based analysis. We saw that the MLP classifier outperformed the FastBDT and GNB classifiers and that the cross-section measurement was more accurate with the MLP classifier than with rectangular cuts. Still, the model we used for it was the most basic model possible. We had constructed a neural network with one hidden layer and a small number of neurons contained in it. There is much room to test more sophisticated neural networks, possibly with many hidden layers and different types of neurons. A natural next step also would be to repeat this analysis on real data. Furthermore, we saw that by combining machine learning models with cut-based analysis, we could get rid of additional background events from the sample. This did not lead to a more precise cross-section measurement however. Here, further investigations are possible, e.g. by observing all variables which the machine learning model uses. The benefit of using the FastBDT classifier lies in the possibility to investigate the variable importance and correlations between variables, which the MLP classifier cannot do. It is also possible to use both methods in conjunction, with the drawback of an increased training time.

While machine learning is being increasingly used in a wide variety of scientific disciplines, it is still a rather new technique in the classification of HEP events, especially involving taus. There is just not enough expertise on this subjects yet. By giving a short but detailed introduction into some machine learning algorithms and then readily applying them to data, we hopefully conveyed at least a little of their usefulness. This is not to say, that the analysis can be fully automated yet. The experimenter still has to choose meaningful variables and an algorithm which fits the problem at hand. But these are tools that the physicist can use in his or her work, and probably will be using more and more in the future.

References

[1] Partice Data Group, official website: *http://pdg.lbl.gov*

[2] Abe, T; Adachi, I; Adamczyk, K et. al.: *Belle II Technical Design Report*, arXiv:1011.0352v1, 2010

[3] Kou, E.; Urquijo, P. ; et. al.: *BELLE II Physics Book*, 2018

[4] BASF2 documentation *https://software.belle2.org*

[5] Nishimura, K.: *The time-of-propagation counter for Belle II*, arXiv:1009.0876v1, 2010

[6] Akai, K.; Furukawa, K.; Koiso, H.: *SuperKEKB collider*, arXiv:1809.01958v2, 2018

[7] Sandilya, S.: *Particle Identification at Belle II*, arXiv:1610.00264v1, 2016

[8] Kuhr, T.; Pulvermacher, C.; Ritter, M., Hauth, T.; Braun, N.: *The Belle II Core Software*, arXiv:1809.04299v2, 2018

[9] James, G.; Witten, D.; Hastie, T.; Tibshirani, R.: *An Introduction to Statistical Learning*, Springer, 2017

[10] Hastie, T.; Friedman, J.; Tibshirani, R.: *The Elements of statistical learning*, Springer, 2017

[11] Raschka, S.: *Python Machine Learning*, packt, 2015

[12] Goodfellow, I.; Bengio, Y.; Courville, A.: *Deep Learning*, mitp, 2018

[13] Rosenblatt, F. : *The perceptron, a perceiving and recognizing automaton*, Cornell Aeronautical Laboratory, 1957.

[14] De La Cruz-Burelo, E.; Salinas, L.; Rad, N. K.; Rostomyan, A.; Hernández-Villanueva, M.: *τ lepton mass measurement at Belle II*, Version 0.1, BELLE2-NOTE-PH-2020-001, April 2, 2020

[15] Anastassov, A.; et.al.: *Experimental tests of lepton universality in τ decay*, 10.1103/PhysRevD.55.2559, March, 1997

List of Figures

List of Tables