МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
"МОСКОВСКИЙ ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ
(ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ)"

ФАКУЛЬТЕТ ИННОВАЦИЙ И ВЫСОКИХ ТЕХНОЛОГИЙ
КАФЕДРА АНАЛИЗА ДАННЫХ

Выпускная квалификационная работа по направлению
01.03.02 «Прикладные математика и информатика»
НА ТЕМУ:

Идентификация заряженных элементарных частиц в данных, набранных в эксперименте LHCb (CERN)

Студент _____ Елтышев Е.Н.

Научный руководитель к.ф-м.н _____ Дольников В.Л.

Консультант к.ф-м.н _____ Деркач Д.А.

МОСКВА, 2016

Содержание

Часть 1

Введение

1.1 Идентификация частиц

В настоящее время экспериментальная физика частиц развивается с огромной скоростью. В мире работает несколько экспериментальных установок, самой большой из которых является Большой адронный коллайдер (БАК). Одной из задач, стоящих перед БАК, является анализ адронных распадов. В рамках экспериментов производится поиск несоответствий теоретической вероятности распада, рассчитанной в Стандартной модели, с экспериментальными измерениями. Поэтому одной из наиболее важных задач является выделение событий, содержащих нужный распад. В некоторых случаях это сделать легко – из-за низкого загрязнения шумом интересующего участка. Однако, некоторые распады имеют очень схожую структуру, как например $B \to DK^{\pm}$ и $B \to D\pi^{\pm}$, и единственным способом различить их является определение продуктов распада. Задача определения типа частицы по отклику детекторов называется идентификация частиц (Particle Identification, PID). Использование PID в анализе распадов играет существенную роль: она позволяет различить близкие распады, улучшает разрешающую способность в массе первичных частиц и снижает количество шума. Определение типа частицы является легкой задачей при известной массе. Главная проблема заключается в том, что масса частицы восстанавливается по отклику детектора с высокой погрешностью, что затрудняет разделение близких по массе частиц, например, B_d и B_s мезонов, отличие в ~ 100 МэВ.

Существует несколько классов детекторов, которые предоставляют важную для идентификации заряженных частиц информацию. Комбинирование этой

информации даёт лучшие результаты, так как каждый класс детекторов работает по своим принципам, например, мюонные камеры хорошо идентифицируют мюоны, но плохо отличают каоны от протонов. Каждый эксперимент проектируется с учётом этих особенностей и основных физических процессов, происходящих на этих энергиях. Эксперимент LHCb содержит два детектора черенковского излучения, электромагнитный и адронный калориметры, мюонные камеры.

В данной работе предлагается построение модели машинного обучения для идентификации частиц. Это позволит комплексно агрегировать информацию с различных субдетекторов и получить универсальную модель, подходящую для анализа разных распадов. Предлагается уделить внимание тщательному анализу признаков, построению новых признаков и комбинированию нескольких моделей.

В качестве базовых алгоритмов идентификации частиц рассматриваются DLL и ProbNN [1]. Первый метод основан на построении вероятностной модели по различным характеристикам частицы, таким как импульс, заряд и геометрическая информация, а также показаниям черенковского субдетектора, электромагнитного и адронного калориметра и мюонных камер. Определение типа частицы производится путем выбора модели с наибольшим правдоподобием. ProbNN является обобщением DLL которое помимо вероятностной модели комбинирует также геометрическую информацию о частице, флаги прохождения через различные слои детектора, положение исходной вершины распада и прочие признаки.

Увеличение точности в задаче идентификации частиц позволит улучшить оценку вероятности распада, что критически важно при анализе сверхредких и запрещённых в Стандартной модели распадов.

1.2 Большой адронный коллайдер

Большой адронный коллайдер представляет собой кольцевой ускоритель заряженных частиц на встречных пучках. Идея постройки данного ускорителя появилась в 1984 году. Строительство БАК началось в 2001 году и заняло 7 лет. БАК находится на территории Швейцарии и Франции, около Женевы.

Коллайдер представляет собой подземный тоннель в виде кольца длиной 26 659 метров, находящийся на глубине 100 метров. Туннель содержит два параллельных канала, пересекающихся в четырех точках. По каждому из каналов курсирует протонный пучок, также существует возможность запускать пучки, состоящие из тяжёлых ядер По всей длине тоннеля расположено более 1200 сверхпроводящих магнитов, которые удерживают пучки внутри установки и около 400 магнитов, отвечающие за фокусировку пучка. Благодаря магнитному полю пучок постоянно поворачивается, оставаясь внутри кольца.

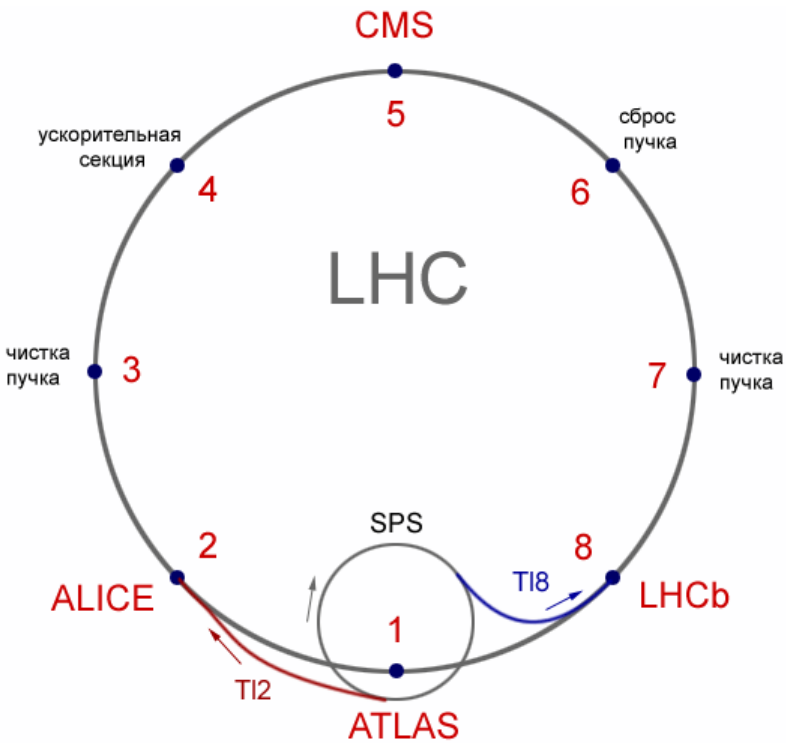

Рис. 1.1: Устройство БАК

Перед подачей в основной ускоритель, сгустки протонов подготавливаются в первичных ускорителях. Первой такой системой является линейный ускоритель LINAC 2, который производит протоны с энергией 50 МэВ. Затем они подаются в Proton Synchrotron Booster (PSB), который разгоняет протоны до энергии в 1.4 ГэВ. Затем пучки протонов подаются на Протонный синхротрон (Proton Synchrotron,

PS), а после этого на Протонный суперсинхротрон (Super Proton Synchrotron, SPS). Они разгоняют пучки до 26 и 450 ГэВ соответственно. И, наконец, из SPS протоны попадают в основной ускоритель, где разгоняются до рабочей энергии в 6.5-7 ТэВ.

Перед Большим адронным коллайдером было поставлено много задач, как например, поиск и анализ характеристик бозона Хиггса, которая позволит пролить свет на механизм формирования массы у частиц. Также на БАК предполагается решить такие задачи как поиск суперсимметрии, изучение топ-кварков и т.д.

Данные задачи решают ученые, объединенные в различные коллаборации. Каждая коллаборация контролирует эксперимент на отдельном детекторе. В настоящее время на БАК работает четыре основных эксперимента: ATLAS, CMS, ALICE и LHCb. Каждый из данных детекторов расположен в одной из 4 точек столкновения пучков. Детектор представляет собой набор регистрирующих устройств, которые собирают информацию о проведенных столкновениях.

Детекторы ATLAS и CMS являются главными экспериментами на БАК. Они построены по классической схеме – в центре расположены трековые детекторы для измерения траекторий частиц, затем – калориметры для измерения энергий, а снаружи – специальные детекторы для регистрации мюонов. Данные детекторы являются многоцелевыми, то есть они могут использоваться для анализа любым процессов с высокоэнергетическими частицами. Детектор ALICE предназначен для изучения столкновений тяжелых ядер (таких как свинец), при столкновении которых рождаются огромное количество отдельных адронов, поэтому критическим требованием к ALICE является способность различать треки отдельных частиц. Детектор LHCb предназначен для изучения свойств адронов, содержащих b-кварк. Такие адроны успевают отлететь от оси пучка на доли миллиметра, поэтому ключевым элементом LHCb является вершинный детектор, который может заметить такое смещение. И в ALICE, и в LHCb важнейшую роль играют системы идентификации частиц.

1.3 Детектор LHCb

1.3.1 Общее описание LHCb

Детектор LHCb (Large Hadron Collider beauty) – это один из четырех основных детекторов, расположенных на Большом адронном коллайдере. Одной из основных задач данного эксперимента является исследование нарушения CP-симметрии в распадах прелестных адронов, где она наиболее сильно проявляется. Несмотря на важность данного явления, оно плохо изучено на сегодняшний день. Анализ причин CP-нарушения позволит ответить на вопрос о различии в количестве материи и антиматерии во Вселенной. Также перед LHCb стоят задачи проверки предсказаний Стандартной модели элементарных частиц и взаимодействий в распадах B-мезонов, изучения распадов очарованных частиц и измерение параметров треугольника унитарности.

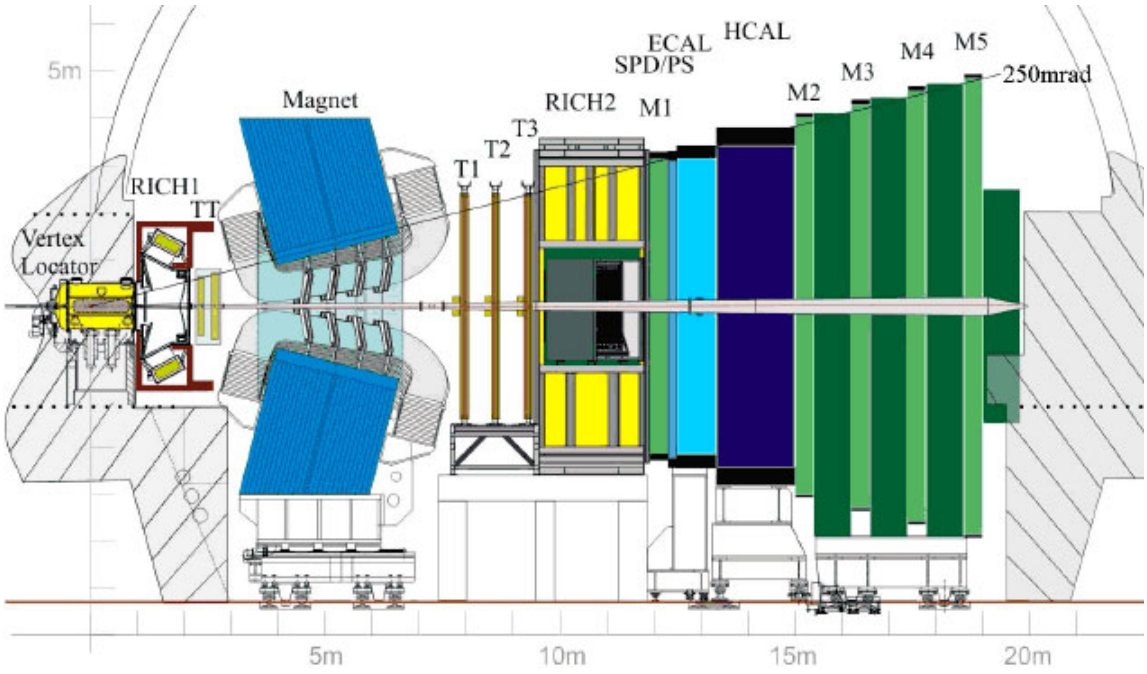

Рис. 1.2: Детектор LHCb

В отличие от других детекторов, которые покрывают точку столкновения с обеих сторон, LHCb имеет вид конуса, на острие которого происходят столкновения встречных пучков. Он может регистрировать лишь частицы, вылетающие близко к оси пучка (не более 15 градусов).

Детектор LHCb имеет классическое многослойное строение. Но так как он имеет вполне конкретные задачи, некоторые части детектора оптимизированы для их решения. Общее устройство детектора показано на рисунке 1.2. Детектор состоит

из 5 основных подсистем (субдетекторов): вершинный детектор VELO, черенковский счетчик RICH, трековые детекторы, электромагнитный и адронный калориметры и мюонные камеры. Горизонтально через все слои проходит вакуумная труба, через которые пролетают пучки протонов.

1.3.2 Вершинный субдетектор VELO

Ближе всего к месту столкновения расположен высокоточный вершинный детектор VELO (Vertex Locator). Он имеет уникальный дизайн, специфичный для задачи. Он состоит из нескольких слоев полупроводниковых плат, расположенных на подвижных кронштейнах внутри вакуумной трубы. VELO имеет два режима работы: когда пучок нестабильный, с помощью кронштейнов пиксельные детекторы уводятся в сторону от оси пучка, чтобы предотвратить его повреждение высокоэнергетичным сгустком протонов. Когда пучок стабилизирован, пиксельные детекторы придвигаются к оси движения протонов на расстояние 5 мм. Благодаря такой близости к точке столкновения, разрешающая способность детектора достигает 10 микрон. Данные платы работают по принципу матрицы цифрового фотоаппарата. Когда заряженная частица пролетает сквозь пластинку, она оставляет на ней след. Этот след считывается электронным элементом. Таким образом, анализируя следы с нескольких последовательных слоев, можно восстановить путь частицы и затем восстановить вершину распада частицы как пересечение нескольких траекторий.

Иногда получается, что таких вершин несколько, причем одна из них лежит на оси пучка, а остальные – нет. Вершина, лежащая на оси, называется первичной, а остальные – вторичными. Это обычно означает, что в первичной вершине столкнулись протоны, породили несколько частиц, которые пролетели некоторое расстояние и распались на другие частицы. Высокая точность восстановления вершины (10 микрон в случае LHCb) позволяет надежно регистрировать случаи, когда вторичные вершины отстоят от основной оси на 100 микрон, что примерно эквивалентно расстоянию, которое успевают пролететь метастабильные адроны, содержащие в себе c- или b-кварки, за время жизни. Как раз такие частицы являются предметом изучения LHCb поэтому вершинный детектор является наиболее важной составляющей детектора LHCb.

1.3.3 Субдетектор RICH

Детектор RICH (Ring Imaging CHerenkov) предназначен для идентификации частиц. Когда заряженная частица двигается в среде со скоростью, превышающей скорость света в данной среде, она порождает вспышку света, форма которой зависит от скорости частицы. Используя массив зеркал, детектор RICH переносит данное излучение на сенсоры, которые измеряют положение и форму вспышки. Затем по форме восстанавливается скорость частицы и, используя геометрическую информацию с вершинного детектора, восстанавливают массу и заряд частицы.

В LHCb RICH детектор состоит из двух частей: первая расположена сразу за вершинным субдетектором (RICH-1), а вторая – за магнитом и трековым субдетектором (RICH-2). RICH-1 заполнен газом C_4F_{10} и кварцевым аэрогелем, что позволяет хорошо измерять импульсы в нижнем диапазоне. RICH-2 содержит газ CF_4; он предназначен для измерения импульсов частиц в среднем и высоком диапазоне.

1.3.4 Магнит

Между RICH детекторами расположен огромный 1600-тонный магнит, имеющий специальную форму для обеспечения однородности магнитного поля. Траектории частиц искривляются в этом поле и с помощью трековых детектором можно измерить радиус кривизны и, следовательно, импульс частицы.

1.3.5 Трековый субдетектор

Трековый субдетектор расположен между магнитом и вторым RICH детектором. Главной его задачей является эффективное восстановление траектории заряженной частицы. Траектории частиц далее используются для определения импульса частицы. Трековая система LHCb состоит из четырех отслеживающих станций (tracking station): одна из них (“TT”) расположена между RICH-1 и магнитом, а три остальных (“T1-T3”) – между магнитом и RICH-2.

Данные станции построены по двум различным технологиям. Первая станция построена по принципу кремниевого трекера, который использует кремниевые микрополосковые линии для определения пролетающих частиц. Заряженные частицы сталкиваются с атомами кремния, высвобождая электроны, которые создают напряжение. Считывая данное напряжение, станция определяет

траекторию частицы. Три остальных станции содержат в себе тысячи наполненных газом дрейфовых трубок. Когда заряженная частица пролетает через газ, возникает ионизация молекул газа, которая производит электроны. Траектория восстанавливается благодаря определению времени, которое требуется электронам для достижения анода, расположенного в центре каждой трубки.

1.3.6 Калориметры

После RICH-2 располагаются калориметры. Они применяются для измерения энергии элементарных частиц. Калориметр содержит внутри себя толстый слой плотного вещества (обычно тяжелого металла). Пролетая сквозь него, частица сталкивается с ядрами атомов или электронами и порождает в результате поток вторичных частиц. Вторичные частицы, в свою очередь, также сталкиваются с атомами вещества и этот процесс порождает ливень из частиц. Через какое-то время ливень останавливается, вторичные частицы поглощаются или аннигилируют и часть энергии выделяется в виде света. Эта вспышка света собирается на торцах калориметра фотоумножителями, которые превращают ее в электрический импульс. Считывая данный импульс можно определить энергию исходной частицы.

Разные частицы порождают ливни из разного вида частиц. Электроны и фотоны в основном сталкиваются с электронными оболочками атомов и порождают ливень из большого числа электронов. Электромагнитные ливни быстро поглощаются в слое вещества толщиной несколько десятков сантиметров. Высокоэнергетические адроны, такие как протоны, нейтроны, пи-мезоны и К-мезоны, сталкиваются по большей части с ядрами атомов вещества. Это порождает адронные ливни, которые проникают гораздо глубже в толщу вещества. Поэтому для поглощения адронного ливня частицы очень высокой энергии требуется один-два метра вещества.

Отличие в поведении электромагнитного и адронного ливней активно используется в современных детекторах. Обычно калориметры имеют два слоя: сначала располагаются электромагнитные калориметры, поглощающие в основном электромагнитные ливни, а затем располагаются адронные калориметры, которые регистрируют адронные ливни. Таким образом, калориметры позволяют определить не только энергию, но и ее происхождение – электромагнитное или адронное. Это различие имеет большую важность при дальнейшей идентификации частиц.

1.3.7 Мюонные камеры

Мюонная камера располагается в самом конце детектора. Принцип ее работы основывается на том, что мюоны очень медленно теряют энергию при движении сквозь вещество. На это есть две причины: с одной стороны, мюоны являются слишком тяжелыми для эффективной передачи энергии электронам при столкновении, а с другой стороны, они не участвуют в сильном взаимодействии, поэтому они слабо рассеиваются на ядрах. Как следствие, мюоны способны пролететь большое расстояние сквозь вещество, проникнув туда, куда не долетают никакие другие частицы.

Такое поведение мюонов, с одной стороны, делает невозможным измерение энергии мюонов в калориметре, а с другой стороны позволяет хорошо отличать мюоны от других частиц. Поэтому в LHCb мюонная камера расположена в самом конце детектора. Ее главная задача измерение не энергии, а импульса мюона и при этом можно с высокой уверенностью считать попавшие в нее частицы мюонами.

Часть 2

Описание задачи

2.1 Постановка задачи

Столкновение частиц встречных пучков протонов называется *событием*. В ходе каждого события рождается большое количество частиц, которые впоследствии распадаются на несколько вторичных частиц. Пролетая сквозь детектор, вторичные частицы оставляют следы в субдетекторах. Данные следы называются *хитами*. Затем по расположению хитов, оставленных частицей, восстанавливается ее путь. Данный путь называется *треком*.

Обычно в событии содержится большое количество треков ($\sim$ 100), поэтому корректное восстановление каждого из них является трудной задачей. При восстановлении трека возможна ситуция, когда несколько треков частиц частично перекрывают друг друга и получается трек, составленный из следов нескольких частиц. Такие треки называются Ghost. В остальных же случаях считается, что трек принадлежит частице одного из следующих типов: {Электрон, Мюон, Пион, Каон, Протон}

Задачей идентификации частиц является определение типа частицы по характеристикам трека.

2.2 Описание признаков

Каждый трек имеет 77 признаков, разбитых на 5 групп: общие характеристики события, характеристики трека (Tracking), данные с субдетектора RICH (RICH), данные с мюонной камеры (Muon) и калориметров (CALO) и комбинированные

правдоподобия частиц (CombDLL).

Характеристики события

Данная группа признаков описывает событие, в котором был порожден данный трек. Каждому событию может принадлежать несколько треков. В данную группу входят следующие признаки:

- NumTTracks - количество треков в событии
- NumUpstreamTracks, NumDownstreamTracks, NumLongTracks - количество треков разного вида
- NumMuonTracks - количество треков в данном событии, содержащих хиты в мюонной камере

Характеристики треков

Данная группа признаков описывает геометрические и физические характеристики трека. В данную группу входят:

- TrackP - продольный импульс
- TrackPt - поперечный импульс
- TrackGhostProbability - оценка вероятности, что данный трек является Ghost
- TrackType - тип трека (Upstream, Downstream, Long)
- TrackChi2PerDof, TrackFitMatchChi2, TrackFitTChi2, TrackFitTNDoF, TrackFitVeloChi2, TrackFitVeloNDoF, TrackMatchChi2, TrackNumDof - количество степеней свободы и коэффициент хи-квадрат по данным вершинного и трекового субдетектора.
- TrackDOCA - расстояние между треком и осью Z

Признаки, полученные в субдетекторе RICH

Для возникновения черенковского излучения необходимо, чтобы импульс частицы был выше определенного значения. Далее для каждого типа частицы,

если импульс выше соответсвующей отсечки, то производится расчет правдоподобия гипотезы, что трек был порожден частицей данного типа.

- RichAboveElThres, RichAboveKaThres, RichAboveMuThres, RichAbovePiThres, RichAbovePrThres - флаг, показывающий был ли импульс больше отсечения или нет.
- RichDLLbt - лог-правдоподобие гипотезы, что "импульс ниже отсечения для всех типов частиц"
- RichDLLe, RichDLLk, RichDLLmu, RichDLLp, RichDLLpi - лог-правдоподобие гипотезы для соответствующего типа частицы.

Подробнее методика расчета DLL признаков приводится в 3.1, а также в [1].

Признаки, полученные в мюонной камеры

- MuonIsMuon - флаг, показывающий, что трек удовлетворяет критериям отбора мюонов.
- MuonIsLooseMuon - флаг, показывающий, что трек удовлетворяет расширенным критериям отбора мюонов.
- MuonMuLL, MuonBkgLL - лог-правдоподобия мюонных гипотез.

Подробнее методика создания данных признаков описывается в [2].

Признаки, полученные в калориметрах

- BremPIDe, PrsPIDe - правдоподобия электронов, полученные с субдетекторов Brem и Prs.
- EcalPIDe, EcalPIDmu - правдоподобия электронов и мюонов, полученные с электромагнитного калориметра.
- HcalPIDe, HcalPIDmu - правдоподобия электронов и мюонов, полученные с адронного калориметра.

Комбинированные правдоподобия

- CombDLLe, CombDLLmu, CombDLLpi, CombDLLk, CombDLLp - комбинированные правдоподобия электронов, мюонов, пионов, каонов и протонов. Подробнее данные переменные описываются в секции 3.1.

2.3 Описание данных

Построение обучающей выборки в задаче идентификации частиц является нетривиальной проблемой. Среди экспериментальных данных с LHCb только небольшое количество частиц можно разметить однозначно, поэтому для создания размеченной выборки применяется моделирование Монте-Карло.

Моделирование Монте-Карло представляет собой эмуляцию работы детектора LHCb, субдетекторов и регистрирующей аппаратуры с использованием суперкомпьютера. Подробнее данный метод моделирования описан в [3, 4].

В результате моделирования было получено N объектов. Для снижения временных затрат на эксперименты, была выбрана случайная подвыборка для обучения размером 1.2 млн. объектов и для контроля размером 1 млн. объектов. После экспериментов по балансировке выборки, описанных в секции 4.1.1, из исходных данных была сгенерирована сбалансированная выборка размером 300 тыс. объектов.

Таблица 2.1: Распределение классов в выборках.

	Обучение	Контроль	Обучение (сбалансированная)
Ghost	170 тыс.	143 тыс.	50 тыс.
Электрон	5.6 тыс	5.2 тыс	50 тыс.
Мюон	30 тыс	41 тыс.	50 тыс.
Пион	775 тыс	640 тыс.	50 тыс.
Каон	140 тыс.	105 тыс.	50 тыс.
Протон	82 тыс.	66 тыс.	50 тыс.

Часть 3

Предшествующие результаты

3.1 Delta log-likelihood модели (DLL)

Пропорции рождаемых при экспериментах частиц являются крайне несбалансированными. Большую часть ($\sim$ 70-80%) составляют пионы. Поэтому при определении типа частицы изначально предполагается, что она является пионом. Затем проверяется набор гипотез, соответствующих другим типам частиц. Поэтому в анализе используется набор относительных лог-правдоподобий:

$$DLL_{electron}(X) = \log \frac{LL_{electon}(X)}{LL_{pion}(X)} = \log LL_{electron}(X) - \log LL_{pion}(X)$$

Информация о типе частицы собирается независимо от мюонной камеры, субдетектора RICH и от калориметров. Каждый из данных субдетекторов имеет собственный набор гипотез что трек порожден той или иной частицей. Идея модели DLL состоит в том, чтобы скомбинировать лог-правдоподобия, полученные от различных субдетекторов в единый набор величин. Комбинированные лог-правдоподобия (CombDLL) имеют вид линейных комбинаций правдоподобий, полученных от RICH, MUON и CALO.

После построения комбинированных правдоподобий производится калибровка модели. По смоделированным данным определяются пороговые значения CombDLL для каждого вида частицы. При превышении порога частица считается, к примеру электроном, а если значение ниже порога - гипотеза электрона отвергается.

Подробнее данная модель описывается в [1].

3.2 ProbNN

Модель ProbNN является развитием подхода CombDLL. Модель DLL делает предположение, что зависимости в данных можно представить в виде линейной комбинации нескольких правдоподобий. Данное предположение сильно ограничивает модель и негативно сказывается на качестве. Также модель DLL требует трудоемкой калибровки: для каждой конкретной задачи требуется определение отсечений.

Для более эффективного использования данных и ослабления предположений о виде модели была предложена модель ProbNN. Помимо признаков DLL, в ней используется геометрическая и физическая информация о треке. ProbNN представляет собой 6 нейронных сетей из библиотеки TMVA[5], построенных по стратегии "один против всех".

Для каждой частицы был отобран свой набор признаков и построена модель один против всех. В качестве модели была использована нейронная сеть TMVA kMLP[5] со следующими параметрами:

- Слои: $K : [1.5 \cdot K] : 1$, где K - количество признаков для данной частицы
- Количество эпох: 200
- Способ обучения: последовательный (без разбиения на батчи)

Результаты данной модели на нашем датасете представлены в таблице 3.1.

Таблица 3.1: Результаты модели ProbNN (ROC AUC)

	ProbNN
Ghost	0.9480
Электрон	0.9827
Мюон	0.9879
Пион	0.9335
Каон	0.9152
Протон	0.9161

Часть 4

Особенности данных и методы их устранения

4.1 Особенности данных

При решении практических задач особое внимание следует обращать на качество данных. Чаще всего они содержат много шума, выбросов и пропусков в данных В нашей задаче можно выделить три основные особенности данных: несбалансированность классов, длинные хвосты распределений некоторых признаков и пропуски в данных. Рассмотрим подробнее каждую особенность.

4.1.1 Несбалансированность классов

Естественные пропорции частиц, получаемых в результате экспериментов, являются крайне несбалансированными. Например, пионы рождаются в 65% случаев, а электроны только в 0.5%. Распределение классов в обучающей выборке можно видеть на рис. 4.1.

Многие методы машинного обучения плохо работают в случае несбалансированности классов. Например, при использовании градиентных методов, параметры модели будут гораздо чаще обновляться по объектам больших классов и реже по объектам маленьких классов. Это приводит к тому, что модель переобучается на больших классах и недообучается на маленьких.

Второй негативный эффект проявляется при использовании подхода «один против всех». Если есть один доминирующий класс (в нашем случае - пион), то при построении модели, к примеру, «электрон против всех», она на самом деле обучается

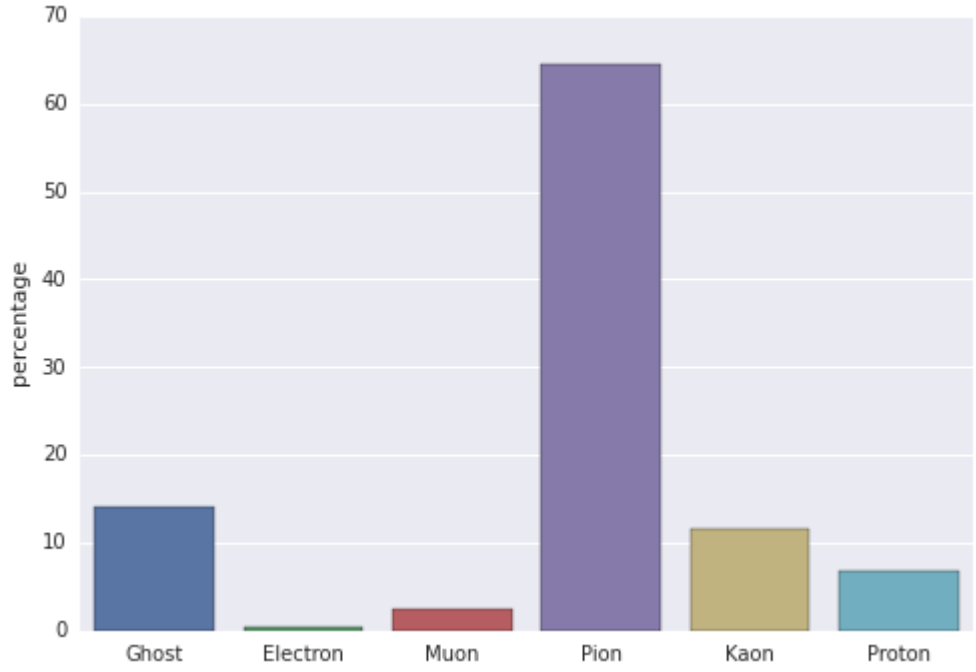

Рис. 4.1: Пропорции классов в обучающей выборке

отличать электроны от пионов.

Тем самым, несбалансированность классов ведет к ухудшению обобщающей способности модели.

Существует несколько методов решения проблемы несбалансированности классов, например прореживание (undersampling) или взвешивание (weighting) выборки [6].

Эксперимент

В рамках эксперимента было рассмотрено два метода: прореживание и взвешивание выборки. Прореженная выборка была получена из той же выборки, что и обучающая, но с применением равномерного выбора. Взвешивание выборки производилось по формуле:

$$w_i = 0.1 \cdot \frac{N}{N_i},$$

где w_i - вес i-того класса, N - размер выборки, N_i - количество объектов i-того класса.

В качестве модели для эксперимента использовалась нейронная сеть Keras со следующими параметрами:

- Количество нейронов по слоям: $77 : 107 : 6$

- Оптимизатор: SGD, темп обучения: 0.01
- Количество эпох: 200

Результаты эксперимента можно видеть в таблице 4.1.

Таблица 4.1: ROC AUC при взвешивании выборки

	Без балансировки	С балансировкой	Прореженная
Ghost	0.9377	**0.9475**	0.9465
Электрон	0.9796	**0.9846**	0.9815
Мюон	0.9874	0.9887	**0.9902**
Пион	**0.9472**	0.9444	0.9449
Каон	0.9094	**0.9197**	**0.9197**
Протон	0.9083	0.9153	**0.9193**

Как можно видеть в таблице 4.1, после балансировки улучшается качество классификации всех частиц, кроме пионов. Так как балансировка производилась с целью снижения переобучения модели под класс пионов, это вполне закономерный результат.

Можно сделать вывод, что оба метода балансировки классов дают сравнимые результаты. Так как в моем распоряжении было достаточное количество данных, было принято решение использовать прореживание выборки как метод балансировки классов.

4.1.2 Длинные хвосты распределений некоторых признаков

Распределение имеет длинный хвост, если оно имеет вид как на рис. 4.3. Наличие длинных хвостов негативно сказывается на работе нейронных сетей, так как наличие больших значений-выбросов ухудшает оценку коэффициентов линейной модели.

В нашем датасете содержится 38 признаков, имеющих длинные хвосты.

Для борьбы с длинными хвостами было применено следующее преобразование:

$$\text{smooth}(x) = \text{sign}(x) \cdot \log(1 + |x|)$$

Данное преобразование обладает многими преимуществами. Во-первых, оно сглаживает распределением таким образом, что значения, близкие к нулю

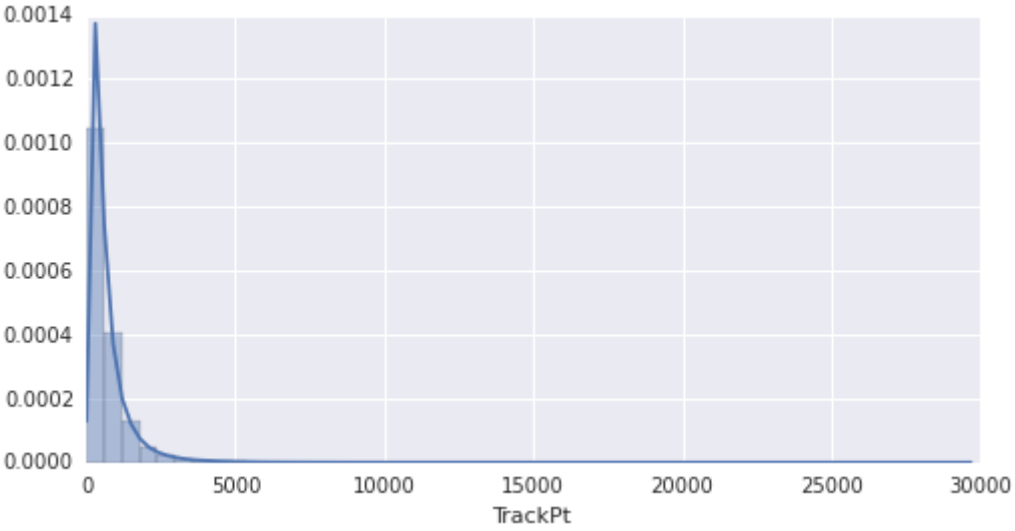

Рис. 4.2: Распределение TrackPt до преобразования

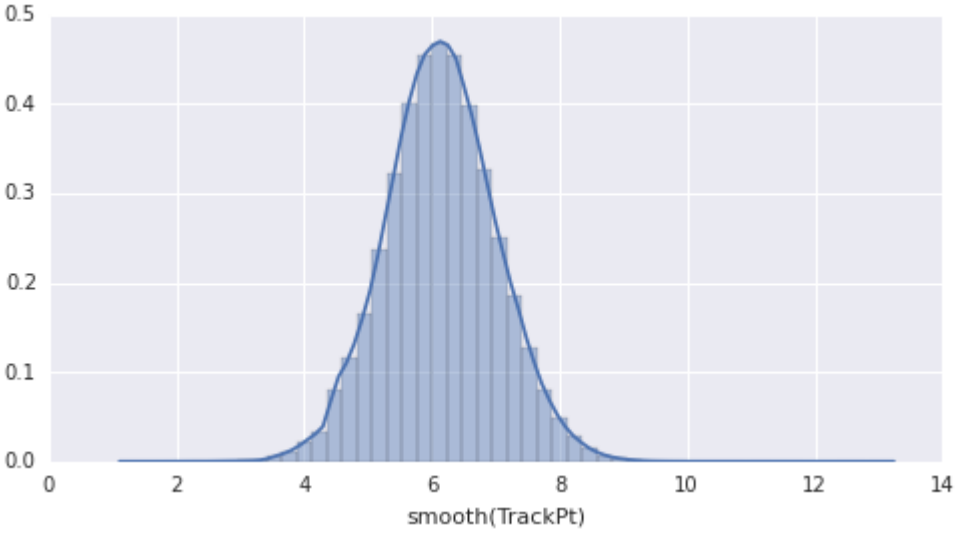

Рис. 4.3: Распределение TrackPt после преобразования

сдвигаются вправо, а большие значения сдвигаются влево. Во-вторых, оно позволяет корректно обработать ситуацию, когда у признака имеется два длинных хвоста влево и вправо. В-третьих, данное преобразование является монотонным, поэтому оно сохраняет относительный порядок объектов, тем самым избегая потери информации.

Вид распределения признака TrackPt можно видеть на рис. 4.3

4.1.3 Пропуски в данных

Проблема работы с пропусками в данных очень распространена в анализе данных [7, 8]. Существует больше количество методов для работы с пропусками в данных, например замена константой, замена на среднее или медиану по признаку или

восстановление значения по другим признакам.

В нашем датасете изначально была произведена замена пропусков на значение -999. Такая замена работает хорошо в методах машинного обучения, основанных на решающих деревьях, потому что построение дерева не зависит от абсолютных значений признаков. Однако, линейные методы крайне чувствительны к масштабу признаков и замена пропусков на большие значения приводит к ухудшению сходимости и оценки коэффициентов линейной модели.

Для работы с линейными методами была произведена предобработка данных. Все пропуски были заменены на среднее значение в колонке, а также для каждого признака с пропусками был добавлен признак-сателлит «был ли признак».

Часть 5

Новые модели

5.1 Логистическая регрессия

Логистическая регрессия - это частный случай обобщенной линейной модели с функцией связи $g(y) = \ln \frac{p}{1-p}$. Данная модель оценивает вероятность принадлежности объекта 2 классам с помощью линейных функций, минимизируя следующий функционал:

$$\frac{1}{2}||w||_2^2 + C\sum_{i=1}^{M} \log\left(1 + \exp(-y_i \cdot (X_i \cdot w + w_0))\right)$$

где w - вектор весов, w_0 - свободный член, X_i - вектор признаков i-того объекта, y_i - метка класса $\{-1, +1\}$, C - коэффициент регуляризации.

Так как логистическая регрессия используется для бинарной классификации, в рамках эксперимента было построено 6 моделей по стратегии «один против всех». Обучение производилось на сбалансированной выборке, тестирование - на контрольной, описанной в пункте 2.3. Перед обучением была произведена предобработка данных, включающая в себя сглаживание распределений, замена пропусков и масштабирование признаков путем вычитания среднего и деления на стандартное отклонение.

Результаты работы логистической регрессии представлены в таблице 5.1. Можно видеть, что логистическая регрессия дает сравнимое качество в 4 классах и значительно уступает в двух (каон и протон).

Таблица 5.1: ROC AUC логистической регрессии

Частица	Baseline	Лог.регр.
Псевдотрек	**0.9480**	0.9436
Электрон	**0.9827**	0.9812
Мюон	0.9879	**0.9886**
Пион	0.9335	**0.9344**
Каон	**0.9152**	0.9043
Протон	**0.9161**	0.8977

5.2 XGBoost

XGBoost - это state-of-the-art библиотека, эффективно реализующая градиентный бустинг[9]. Идея бустинга [10] состоит в последовательном построении взвешенной суммы базовых алгоритмов:

$$f(x) = \sum_{i=1}^{M} \alpha_i h_i(x)$$

Базовые алгоритмы подбираются таким образом, чтобы последующие модели компенсировали ошибки предыдущих. Классический вариант бустинга AdaBoost использовал экспоненциальную функцию в виде функции потерь:

$$L(f(x_i), y_i) = \exp(-y_i \cdot f(x_i))$$

Градиентный бустинг обобщает данную идею на произвольную функцию потерь [11]. Идея градиентного бустинга состоит в моделировании i-тым членом композиции антиградиента функции потерь для композиции из i-1 алгоритма. Это позволяет использовать любую дифференцируемую функцию в качестве функции потерь.

В последнее время широкое распространение получил градиентный бустинг над решающими деревьями [12]. Использование решающих деревьев имеет несколько преимуществ: малая чувствительность к выбросам, способность эффективно работать с пропусками в данных и поддержка многоклассовой классификации без использования подхода «один против всех». В то же время, градиентный бустинг подвержен таким недостаткам, как чувствительность к несбалансированности выборки и шуму в данных.

В эксперименте с XGBoost была использована сбалансированная выборка.

Таблица 5.2: Параметры модели XGBoost

Параметр	Значение
objective	multi:softprob
eta	0.1
max_depth	5
num_class	6
num_boost_round	100

Таблица 5.3: ROC AUC модели XGBoost

Частица	Baseline	XGBoost
Псевдотрек	0.9480	**0.9553**
Электрон	0.9827	**0.9880**
Мюон	0.9879	**0.9936**
Пион	0.9335	**0.9521**
Каон	0.9152	**0.9281**
Протон	0.9161	**0.9256**

Качество проверялось на контрольной выборке, описанной в пункте 2.3. На этапе предобработки, пропуски в данных были заменены на -999. Вместо обучения 6 моделей «один против всех», в данном случае была обучена одна многоклассовая модель.

Параметры модели представлены в таблице 5.2. Параметры, которые не указаны в данной таблице были взяты равными значениям по умолчанию. Подробно параметры модели описаны в [13].

Результаты работы модели XGBoost представлены в таблице 5.3. Можно видеть, что данная модель превосходит базовое решение по всем показателям.

5.3 Нейронные сети

Нейронные сети получили широкое распространение в последнее время. Стремительное увеличение вычислительной мощности современных компьютеров позволяет обучать нейронные сети больших размеров на массивных датасетах.

Целью использования нейросети в данной задаче было сравнение библиотеки

Таблица 5.4: ROC AUC нейронной сети.

Частица	Baseline	Нейросеть
Псевдотрек	0.9480	**0.9531**
Электрон	0.9827	**0.9836**
Мюон	0.9879	**0.9920**
Пион	0.9335	**0.9484**
Каон	0.9152	**0.9178**
Протон	**0.9161**	0.9150

TMVA с возможностями современных библиотек нейронных сетей. Для экспериментов была использована библиотека Keras [14, 15].

Нейронная сеть из базового решения имела несколько недостатков. Во-первых, она обучалась примерно на половине признаков (для каждой из 6 моделей был выбран свой набор признаков). Во-вторых, базовые модели обучались в последовательном (sequential) режиме, без разбиения выборки на батчи. Эти факторы негативно сказывались на качестве классификации.

В отличии от базового решения, данная модель обучалась на всех признаках, а данные на вход подавались в пакетном (batch) режиме. Обучение производилось на сбалансированном датасете, проверка качества - на контрольной выборке, описанной в пункте 2.3. Преобработка данных включала в себя сглаживание распределений, замена пропусков и масштабирование данных.

Архитектура нейронной сети имела следующий вид:

- Слои: $77 : 107 : 6$ нейронов.
- Функция активации скрытого слоя - сигмоида.
- Сеть обучалась с помощью стохастического градиента с темпом обучения 0.01.
- Размер батча - 128 объектов.

Результаты данной модели представлены в таблице 5.4. Можно видеть, что данная нейронная сеть превосходит базовое решение в 5 из 6 случаев.

5.4 Специальная структура нейронной сети

Как было описано в пункте 2.2, в данной задаче признаки можно разделить на несколько групп. Идея данной модели состоит в том, чтобы эффективно учесть взаимодействия признаков внутри группы. Для этого была спроектирована архитектура нейронной сети специального вида, представленного на рис. 5.1. Данные подаются на вход группами, первый слой разбит на блоки, причем связи между нейронами есть только внутри одного блока. Затем все блоки объединяются все последующие слои являются обычными полносвязными слоями.

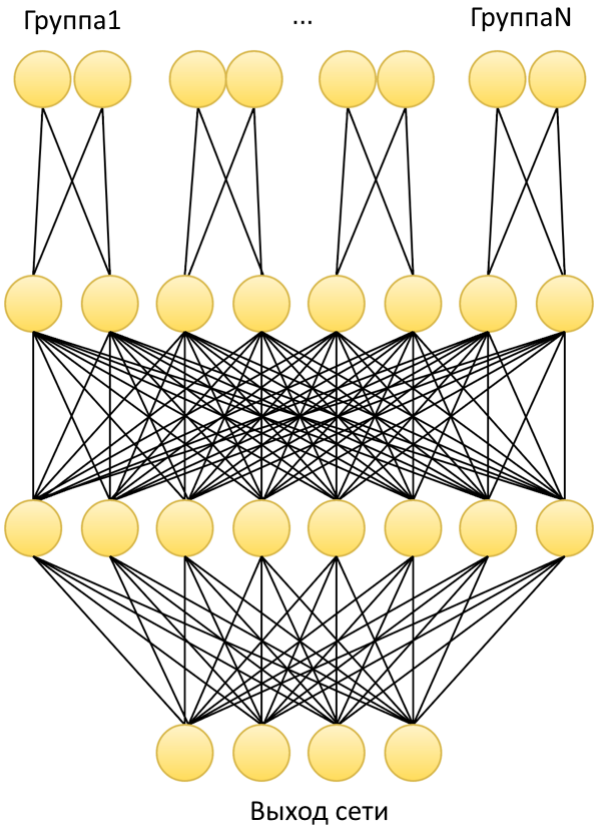

Рис. 5.1: Блочная архитектура нейросети.

В рамках экспериментов было рассмотрено два варианта разделения на блоки: в первом случае признаки делились на блоки согласно принадлежности субдетектору (табл. 5.6), во втором случае признаки делились на три группы - бинарные, DLL и остальные (табл. 5.7).

Результаты представлены в табл. 5.5. Как можно видеть, вариант с разделением признаков на три группы показывает лучшее качество в случае электронов. В остальных случаях нейронная сеть с блочной архитектурой показывает качество хуже, чем нейронная сеть стандартной архитектуры.

Таблица 5.5: ROC AUC нейронной сети со стандартной и блочной архитектурой.

Частица	Baseline	NN	BlockNN 3 группы	BlockNN группы по субдетекторам
Псевдотрек	0.9480	**0.9531**	0.9500	0.9425
Электрон	0.9827	0.9836	**0.9849**	0.9842
Мюон	0.9879	**0.9920**	0.9918	0.9897
Пион	0.9335	**0.9484**	0.9477	0.9418
Каон	0.9152	**0.9178**	0.9038	0.8995
Протон	**0.9161**	0.9150	0.9104	0.9047

Плохие результаты работы данного подхода указывают на то, что взаимодействия внутри выделенных групп вносят незначительный вклад, в то время как взаимодействие признаков из разных групп значительно влияет на качество. Возможно стоит проанализировать другие варианты разбиения признаков на группы.

Таблица 5.6: Вариант разбиения на блоки по принадлежности к различным субдетекторам

Группа	Признаки
Acceptance	InAccBrem, InAccEcal, InAccHcal, InAccPrs, InAccSpd, InAccMuon
CALO	BremPIDe, PrsPIDe, EcalPIDe, HcalPIDe, EcalPIDmu, HcalPIDmu, CaloChargedEcal, CaloChargedPrs, CaloChargedSpd, CaloBremMatch, CaloEcalE, CaloElectronMatch, CaloHcalE, CaloPrsE, CaloSpdE, CaloTrMatch, CaloTrajectoryL, CaloNeutralEcal, CaloNeutralPrs, CaloNeutralSpd
RICH	RichAboveElThres, RichAboveKaThres, RichAboveMuThres, RichAbovePiThres, RichAbovePrThres, RichDLLbt, RichDLLe, RichDLLk, RichDLLmu, RichDLLp, RichDLLpi
Muon	MuonBkgLL, MuonMuLL, MuonNShared, MuonIsLooseMuon, MuonIsMuon
DLL	RichDLLpi, RichDLLe, RichDLLp, CombDLLmu, CombDLLpi, CombDLLe, CombDLLk, RichDLLbt, CombDLLp, RichDLLk, RichDLLmu
Track	TrackChi2PerDof, TrackDOCA, TrackFitMatchChi2, TrackFitTChi2, TrackFitTNDoF, TrackFitVeloChi2, TrackFitVeloNDoF, TrackGhostProbability, TrackNumDof, TrackP, TrackPt

Таблица 5.7: Вариант разбиения на блоки №2

Группа	Признаки
Бинарные	RichDLLpi, MuonIsLooseMuon, InAccSpd, CaloSpdE, NumCaloHypos, RichAboveElThres, CaloChargedSpd, InAccPrs, RichUsedR1Gas, InAccHcal, RichAbovePiThres, InAccEcal, RichAbovePrThres, RichAboveKaThres, CaloNeutralSpd, CombDLLpi, InAccBrem, RichAboveMuThres, MuonIsMuon, InAccMuon, RichUsedR2Gas
DLL	RichDLLpi, CombDLLmu, CombDLLe, CombDLLk, CombDLLp, CombDLLpi, RichDLLbt, RichDLLk, RichDLLe, RichDLLmu, RichDLLp
Остальные	Все остальные

Часть 6

Заключение

В данной работе было рассмотрено решение задачи идентификации частиц с помощью методов машинного обучения. Были проанализированы особенности данных и предложены методы их устранения. В частности, несбалансированность данных была устранена путем генерации сбалансированной подвыборки, было произведено сглаживание распределений некоторых признаков, а также были устранены пропуски в данных. Помимо предобработки, были рассмотрены новые модели для классификации: логистическая регрессия, градиентный бустинг и нейронные сети. Базовое решение было улучшено на 12% в случае нейронной сети и на 25% в случае XGBoost.

6.1 Дальнейшие исследования

Дальнейшие исследования могут быть направлены по двум направлениям. Во-первых, физическая постановка задачи накладывает ограничение равномерности по импульсу на модель. Это означает, что модель должна показывать одинаковое качество при различных значениях импульса и поперечного импульса. Равномерность модели ведет к снижению систематической погрешности вычислений и позволяет более точно оценивать параметры распадов.

Во-вторых, полученные модели могут быть использованы для работы в онлайн режиме. Важным этапом регистрации событий на LHCb является преселекция. Входящий поток данных анализируется и производится отсев неинформативных событий. Хотя большая часть событий отсеивается на аппаратном уровне, на программный уровень события поступают со скоростью порядка 10 тыс. событий в секудну. Это накладывает требования высокой производительности на модель. В

качестве базовой модели для данного случая можно использовать нейронные сети в силу их высокого быстродействия.

Литература

[1] LHCb collaboration, R. Aaij, B. Adeva, M. Adinolfi, A. Affolder, Z. Ajaltouni, S. Akar, J. Albrecht, F. Alessio, M. Alexander, and et al. LHCb Detector Performance. *ArXiv e-prints*, December 2014.

[2] F Archilli, X Cid Vidal, JA Hernando Morata, G Lanfranchi, JH Lopes, M Palutan, E Polycarpo, A Sarti, and B Sciascia. Muon identification performance at lhcb with the 2010 data. 2011.

[3] Torbjorn Sjostrand, Stephen Mrenna, and Peter Skands. Pythia 6.4 physics and manual. *JHEP*, 05:026, 2006.

[4] Torbjörn Sjöstrand, Stephen Mrenna, and Peter" Skands. A brief introduction to PYTHIA 8.1. *Comput.Phys.Commun.*, 178:852–867, 2008.

[5] A. Hoecker, P. Speckmayer, J. Stelzer, J. Therhaag, E. von Toerne, H. Voss, M. Backes, T. Carli, O. Cohen, A. Christov, D. Dannheim, K. Danielowski, S. Henrot-Versille, M. Jachowski, K. Kraszewski, A. Krasznahorkay, Jr., M. Kruk, Y. Mahalalel, R. Ospanov, X. Prudent, A. Robert, D. Schouten, F. Tegenfeldt, A. Voigt, K. Voss, M. Wolter, and A. Zemla. TMVA - Toolkit for Multivariate Data Analysis. *ArXiv Physics e-prints*, March 2007.

[6] H. He and E. A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, Sept 2009.

[7] Bradley Efron. Missing data, imputation, and the bootstrap. *Journal of the American Statistical Association*, 89(426):463–475, 1994.

[8] John W Graham, Patricio E Cumsille, and Elvira Elek-Fisk. Methods for handling missing data. *Handbook of psychology*, 2003.

[9] Tianqi Chen and Tong He. xgboost: extreme gradient boosting. *R package version 0.4-2*, 2015.

[10] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[11] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[12] Jerome H Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378, 2002.

[13] Xgboost parameters. `https://github.com/dmlc/xgboost/blob/master/doc/parameter.md`. Accessed: 2016-06-13.

[14] Keras: Deep learning library for theano and tensorflow. `http://keras.io/`. Accessed: 2016-06-13.

[15] Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, David Warde-Farley, and Yoshua Bengio. Theano: new features and speed improvements. *CoRR*, abs/1211.5590, 2012.