Charm-Jet Physics with Machine Learning

A thesis submitted in fulfilment of the Award of the degree of

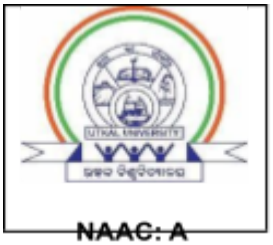
MASTER OF TECHNOLOGY
IN
INFORMATION TECHNOLOGY

By

Bibhu Prasad Mohanty
Roll No – 412VMIT21006
Batch-(2021-23)

Under the guidance of

Dr. Sonia Parmar
PhD, Panjab University and CERNw



CENTRE FOR IT EDUCATION
M. TECH. IN INFORMATION TECHNOLOGY

P. G. DEPT. OF STATISTICS, UTKAL UNIVERSITY, VANI VIHAR,
BHUBANESWAR-751004

CENTRE FOR IT EDUCATION

M. TECH. IN INFORMATION TECHNOLOGY

P. G. DEPT. OF STATISTICS, UTKAL UNIVERSITY, VANI VIAHR,
BHUBANESWAR-751004

.....

CERTIFICATE

This is to certify that **Bibhu Prasad Mohanty**, a bona fide student of M.Tech (IT) bearing **Roll No. 412VMIT21006**, has submitted the thesis entitled "**Charm-Jet Physics with Machine Learning"** in fulfilment of the requirement for the degree of Master of Technology in Information Technology of Utkal University.

Prof. P. R. Dash

Course Director,

M.Tech (IT), Utkal University

CERTIFICATE

This is to certify that the thesis entitled **"Charm-Jet Physics with Machine Learning"** submitted by **Bibhu Prasad Mohanty**, Roll No.-**412VMIT21006**, in fulfilment of the requirements for the Master of Technology in INFORMATION TECHNOLOGY is a bona fide piece of work carried out by him under my supervision and guidance.

To the best of my knowledge, this work has not been submitted to any other University or Institution for the award of any degree. In my opinion, the thesis fulfils the requirement of the regulations relating to the nature and standard of work for Master of Technology.

Dr. Sonia Parmar

PhD, Panjab University and CERN

CERTIFICATE OF APPROVAL

This is to certify that we have examined the thesis entitled **"Charm-Jet Physics with Machine Learning"** submitted by **Bibhu Prasad Mohanty** bearing **Roll No. 412VMIT21006**. We hereby accord our approval of the thesis work carried out and presented in a manner required for its acceptance for the fulfilment of the Degree of Master of Technology in Information Technology. It is understood that this approval does not necessarily endorse or accept every statement made, opinion expressed, or conclusions drawn as recorded in this thesis. It only signifies the acceptance of the thesis for the purpose for which it has been submitted.

(External Examiner)

DECLARATION

I hereby declare that the subject matter of this thesis entitled **"Charm-Jet Physics with Machine Learning"** is the record of work done by me. To the best of my knowledge and belief, the content of this thesis has not been submitted in any institution for the award of any previous higher degree. That the work done by me is free from plagiarism. This is being submitted to Utkal University for which it is required.

Bibhu Prasad Mohanty

Roll No. 412VMIT21006

ACKNOWLEDGEMENTS

Completing the thesis of Master of Technology in Information Technology is an achievement for me and this would not have happened without support from the people around me. Firstly, I would like to express my gratitude towards my guide Dr. Sonia Parmar for her immense knowledge, understanding, support and encouragement to complete my thesis. Her faith in my potential made me much stronger in handling technical challenges. She helped me to overcome various difficulties that I have faced at various stages during this work. I would like to sincerely thank Prof. P. R. Dash, Course Director of M. Tech. (IT), Utkal University for providing all necessary facilities that led to the successful completion of my thesis. I would like to thank the Faculty Members of M. Tech. (IT), Utkal University for their direct and indirect support. I must acknowledge the academic resources which I have collected from various sources. I would like to especially thank and acknowledge my brother Auro Prasad Mohanty for all the guidance provided by him in completing this work. Finally, I shall be forever indebted to my Godly parents, family members and friends for their encouragement, support, and understanding when it was most required.

Bibhu Prasad Mohanty

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The study of particle physics aims to understand the fundamental nature of matter and the universe. Scientists study the tiniest building blocks of the universe by smashing the protons at very high speed. These high-energy collisions of protons provide an excellent laboratory for investigating the behaviour of such subatomic particles like quarks and gluons which are indivisible fundamental particles and cannot be broken down into smaller components. The basic building blocks of matter consists of fundamental particles categorised into three generations. The quarks exist in 6 flavors named; up (u), down (d), charm (c), strange (s), top (t), and beauty (b). The other fundamental particles called leptons are, electron (e), muon (μ), and tau (τ) along with their respective neutrinos electron neutrino (ν_e), muon neutrino (ν_μ), and tau neutrino (ν_τ). The strong force, one of the four main forces in the universe (gravitational force, weak force and electromagnetic force), is what binds quarks together to form protons, neutrons residing within atomic nuclei. Additionally, this compelling force binds quarks to form other fascinating particles referred to as “hadrons". The gluon (g) is a force carrier for the strong force. Similarly, photon is a force carrier for the electromagnetic interaction, (W) and (Z) bosons are responsible for weak interactions. The discovery of Higgs boson (H) completed the theory describing the fundamental forces except gravity. The comprehensive investigations of these elementary particles and their interplay helps to gain insights into the fundamental forces that shape the universe’s very fabric. These interactions between particles offer a profound understanding of the early universe by shedding light on the information about how the universe works and how it began.

In laboratories worldwide, cutting-edge technologies are deployed to accelerate the particles to such a high energies those were previously unimaginable, enabling scientists to probe ever deeper into the mysteries of particle physics. One of such laboratories, has been built in Geneva, Switzerland. The world’s largest accelerating machine, LHC (Large Hadron Collider) [1] is located at CERN which collides beams of different particles such as, p-p, p-Pb, Pb-Pb and Xe-Xe. The LHC has four major experiments (CMS, ATLAS, ALICE, and LHCb) which focuses on different physics goals. For example, the CMS and the ATLAS discovered a new particle in 2020 known as Higgs. It was one of the major discoveries of the LHC. The ALICE is a dedicated heavy-ion detector aims

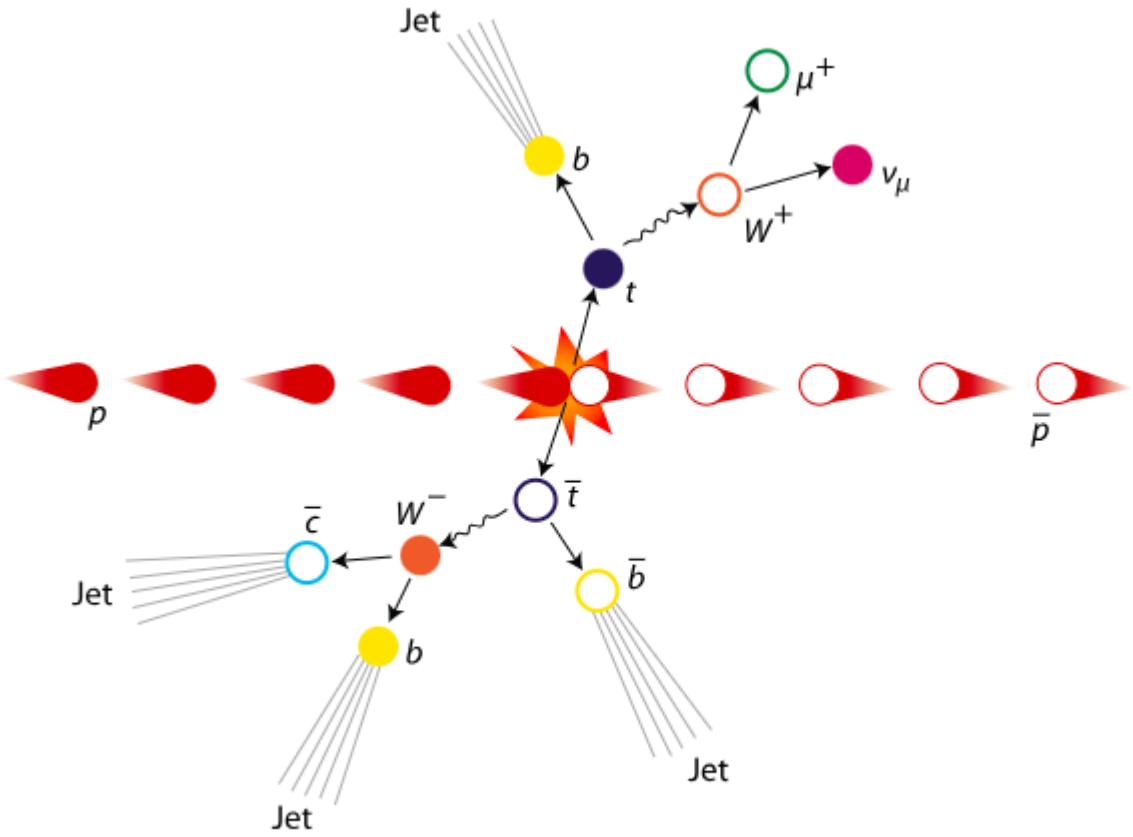


FIGURE 1.1: Proton–antiproton beams smashing into each other producing new particles and jets.

to understand the formation of the universe, i.e., the study of the Quark–Gluon Plasma (QGP).

1.1 Jets

The LHC produces a large number of particles in each collision. The particles flying in all directions (covering angle from 0° to 360°) are detected by the detector. For this thesis, the data are generated using a PYTHIA event generator. PYTHIA also records particles flowing in all directions exactly in a similar way as the LHC experiments do.

Jets [2], which are sprays of particles result from the breakup of quarks and

gluons, are particularly useful for understanding the strong interaction. Scientists have studied jets extensively in particle physics experiments, both at colliders like the LHC and in other experiments. Figure 1.1 represents the proton-proton beams coming from opposite ends smashing into each other. The collision produces a large number of particles which are further hadronized to composite particles called hadrons. The produced particles are represented by different colours along with the jets which are also labelled in figure.

The charm quarks, are of particular interest due to their large masses, also known as “heavy quarks". They can significantly affect the properties of jets as they are produced. In this thesis, we are interested in studying the charm hadron jets containing at least one charm quark as a constituent in each reconstructed jet. These jets are reconstructed using an anti-kt algorithm provided by fastjet simulations package provided by the FastJet package [3].

This thesis studies the proton-proton collision data produced by the PYTHIA event generator at centre of mass energy 7 TeV for the charm jet physics.

1.2 Thesis Organization

Chapter 1 introduces the basics of high energy physics and the charm physics. The tools used to generate the PYTHIA events are discussed in Chapter 2. Chapter 3 includes an introduction of machine learning models and their implications. The analysis strategy and results are discussed in Chapter 4. Finally, Chapter 5 concludes the thesis.

1.3 Objective

The main objective of this thesis is to study charm-jet physics using machine learning techniques. Specifically, we aim to:

- **Event generation:** Generate proton-proton collision events using Monte Carlo event generator PYTHIA.
- **Jet reconstruction:** Reconstruct charm jets using the jet reconstruction algorithm named "anti-kt" and identifying charm hadron jets where each jet must contain at least a charm quark.
- **Machine Learning:** Use machine learning techniques to analyse the properties of charm jets and explore their behaviour in different collision scenarios.

Chapter 2

Monte Carlo Event Generator: PYTHIA

PYTHIA [4] is a powerful tool widely used in high-energy physics research for simulating particle collisions and generating Monte Carlo event samples. It is specially designed for simulating proton-proton collisions at very high energies. The input to PYTHIA includes the parameters of the colliding particles (e.g., electrons, protons, photons, and heavy nuclei), such as the centre-of-mass energy ($\sqrt{s}$), different stages of the collision, hard and soft interactions, fragmentation and decay of produced particles. PYTHIA then generates a series of events that simulate the different possible outcomes of the collision. PYTHIA is categorised as a general purpose Monte Carlo event generator.

2.1 PYTHIA

For this thesis, proton-proton collision events have been generated at centre of mass energy 7 TeV. By using different aspects of proton-proton collisions such as, properties of different particles, correlation between different particles, the energy and momentum of the produced particles, one can do a nice research. We are interested in studying the physics of charm hadron jets produced in high energy collisions to understand the evolution of the universe.

2.1.1 Chromo

To proceed further, first step is to generate proton-proton events using PYTHIA. For that matter, chromo (Cosmic ray and HadROnic interactiOn MOnte-carlo frontend) package is used. It provides a simple and generic user interface to popular event generators used in high energy physics. Chromo simplifies the simulation of particle interactions and make it easier and faster to use. One can start using PYTHIA event generator just by importing the required library as,

```
1 from chromo.models import Pythia8
```

The other important tools used are:

- Hard QCD process to generate the jet events with charm quark as a constituent quark. It is written as: "HardQCD:gg2ccbar = on". Where g represents "gluon", c and $\bar{c}$ represents "charm" and anticharm quark respectively.
- Provide the seed value (=45; in our case) to generate the random events.
- PYTHIA generates events with a tag "status" which represents the state of particles as initial, intermediate or final state. The status is set to 1 to proceed with final state particles only.
- Also the particle identification values for all the charm hadron are given to be helpful in the reconstruction process as follows.

```
    charm_hadrons = {
         "D0Meson": 421,
         "DPlusMeson": 411,
         "DstarPlusMeson": 413,
         "DStragePlusMeson": 431,
         "Dstar0Meson": 423,
         "LambdaCharm+": 4122,
         "LambdaCharm_2595+": 14122,
         "LambdaCharm_2625+": 104122,
         "LambdaCharm_2880+": 204126,
         "SigmaCharm_2455_0": 4112,
         "SigmaCharm_2455+": 4212,
         "SigmaCharm_2455++": 4222,
         "SigmaCharm_2520_0": 4114,
         "SigmaCharm_2520+": 4214,
         "SigmaCharm_2520++": 4224,
        }
```

The D0-meson representing a parent particle decays into its daughter particles kaon and pion. Along with this kaon-pion pair, several othere kaons and pions have been produced in an event as a final state particle. So to differentiate and to include the kaon-pion pair originating from the D0-meson into a single jet, the daughters are replaced with their parents. In this case, kaon-pion pair is replaced with D0-meson. Similar procedure is followed for all other charm hadrons before going to the next step of jet reconstruction.

- The properties of the final state particles such as px, py, pz, E, pt, eta, phi, and pid, produced in each event are stored in a ROOT file format using uproot library used to read and write ROOT files in pure python format.

```
    import uproot
```

has been used to load the uproot library. The same ROOT file has been used further for the reconstruction of charm hadron jets.

2.1.2 Fastjet

Fastjet [3] is a library for performing Jet-Finding within the Scikit-HEP ecosystem. The library includes the classic interface, and a new interface built to perform clustering on multi-event Awkward Array objects [5]. To start with fastjet, one need to install it with a single command as *pip install fastjet* and import library for jet reconstruction

```
import fastjet._pyjet
```

Fastjet library needs some specific information to reconstruct the jet. It is written as follows.

- Declare the radius (R) of the jet. In our case we have set the R value to 0.7 implies that the reconstructed jet must have the radius of R = 0.7.
- Define the jet by providing the name of the algorithm used for jet reconstruction and the value of R. The one used in this thesis is antikt algorithm. It is written as.

```
    jet_def = fastjet.JetDefinition(fastjet.antikt_algorithm,
   R)
```

- Next step is to make the clusters by feeding the particle array along with the jet definition as.

```
    cs = fastjet._pyjet.AwkwardCluterSequence(particle\_array,
    jet\_def)
```

- Make a list of the raw jets just created in the previous step.

```
    raw_jets = cs.inclusive_jets().to_list()
```

Finally the jets have been reconstructed in each event and are stored in a list named "raw_jets". The list contains jet information of four momenta (p_x, p_y, p_z, E). The transverse momentum (p_t), pseudorapidity (η), azimuthal angle (ϕ), and invariant mass (m) of the jets have been calculated using jet four momenta. These are the *"independent features"* extracted from the jets to classify them using machine learning models.

2.2 Finding charm hadron jets

All reconstructed jets in each event do not contain charm hadrons. Thus one needs to find the jet containing charm hadron. In order to do that, the η and ϕ position of each jet is required as well as for all the charm hadrons. In order to find the charm jet, it has to be in the radius less than 0.7, i.e., the jet radius. The charm jet outside the radius

The formula used to find the charm jet is,

$$\Delta R = \sqrt{(\delta_\eta^2 + \delta_\phi^2)} \tag{2.1}$$

where,

$\delta_\phi = \phi_{charm}$ - ϕ_{jet} and $\delta_\eta = \eta_{charm}$ - η_{jet}

The closest charm jet in each event will be of the radius less than the jet radius (R). The *"dependent variable"* is created by labelling as 0 and 1 representing the non-charm and charm hadron jets for each reconstructed jet.

All the independent features and dependent variable are stored in a ROOT file and is further transformed into a data frame to run the machine learning models.

Chapter 3

Machine Learning

Machine learning [6] techniques have become increasingly important in particle physics research. These techniques allow for the analysis of large datasets and complex physical phenomena, and have been used to identify particles, classify events, and improve the accuracy of theoretical predictions. Machine learning involves developing algorithms and different statistical models that enable computers to automatically learn patterns from the data [7]. Based on the newly learned knowledge from existing data, they can become more accurate at predicting new outcomes. In many aspects, machine learning is present in our day-to-day life, for example: recommendations we receive on YouTube channels while watching any program, as well as on online shopping platforms, and voice assistants on our smartphones. Machine learning is useful in almost every field, such as finance for fraud detection, health care for disease diagnosis, transportation for traffic prediction etc. In this thesis, we use machine learning techniques to classify the charm hadron jets produced in proton-proton collision data.

3.1 Classification

Classification is one of the fundamental tasks in machine learning (ML). The goal is to predict the category or class label of a given input based on the patterns and features present in the data [7]. Classification models are used in various domains, such as image recognition, category prediction, spam detection, medical diagnosis and many more. The commonly used classification models are:

3.1.1 Logistic Regression

Logistic regression [8] is a statistical model used to analyse the relationship between a dependent variable and one or more independent variables (often denoted as X), where the dependent variable (denoted as y, i.e., class label) is categorical or binary in nature. In other words, it is a technique used to model and predict the probability of an event occurring, based on the values of one or more independent variables. In logistic regression, the dependent variable is

represented by a binary variable that takes on one of two values, typically 0 or 1. The relationship between the dependent variable and the independent variables is modelled using a logistic function. Mathematically, the logistic regression is defined as,

$$p(z) = \frac{1}{(1 + exp^{-z})} \tag{3.1}$$

where z is a linear combination of input features $(x1, x2, ..., xn)$ and their associated weights $(w1, w2, ..., wn)$, plus an intercept (c) term. It is written as,

$$z = c + w1 * x1 + w2 * x2 + ... + wn * xn \tag{3.2}$$

The logistic function transforms the values of the independent variables into a probability score belongs to class 1. The probability of belonging to class 0 can be obtained as $1 - p(z)$.

The goal of logistic regression is to determine the coefficients of the independent variables that best predict the value of the dependent variable, by estimating the probability of the dependent variable taking on the value 1, given the values of the independent variables. Logistic regression is widely used in various fields such as medicine, biology, marketing, and many others, where the goal is to analyse and predict the probability of an event occurring, based on one or more independent variables.

3.1.2 Support Vector Machine

Support Vector Machine (SVM) [9] is a powerful machine learning algorithm used for linear or nonlinear classification, regression, and even outlier detection tasks. SVMs can be used for a variety of tasks, such as text classification, image classification, spam detection, handwriting identification, gene expression analysis, face detection, and anomaly detection. SVMs are adaptable and efficient in a variety of applications because they can manage high-dimensional data and nonlinear relationships. SVM algorithms are very effective as we try to find the maximum separating hyperplane between the different classes available in the target feature. Support Vector Machine (SVM) is a supervised machine learning algorithm used for both classification and regression. Though we say regression problems as well it's best suited for classification. The main objective of the SVM algorithm is to find the optimal hyperplane in an N-dimensional space that can separate the data points in different classes in the feature space. The hyperplane tries that the margin between the closest points of different classes should be as maximum as possible. The dimension of the hyperplane depends upon the number of features. If the number of input features is two, then the hyperplane is just a line. If the number of input features is three, then the hyperplane becomes a 2-D plane. It becomes difficult to imagine when the number of features exceeds three.

Consider a binary classification problem with two classes, labelled as +1 and -1. We have a training dataset consisting of input feature vectors X and their corresponding class labels y.

The equation for the linear hyperplane can be written as:

$$w^T x + b = 0 \tag{3.3}$$

The vector w represents the normal vector to the hyperplane. i.e the direction perpendicular to the hyperplane. The parameter b in the equation represents the offset or distance of the hyperplane from the origin along the normal vector w.

The distance between a data point x_i and the decision boundary can be calculated as:

$$d_i = \frac{w^T x_i + b}{||w||} \tag{3.4}$$

where $||w||$ represents the Euclidean norm of the weight vector w.

For Linear SVM classifier :

$$\hat{y} = \begin{cases} 1 : w^T x + b \geq 0 \\ 0 : w^T x + b < 0 \end{cases} \tag{3.5}$$

3.1.3 k Nearest Neighbors Classifier

The k-Nearest Neighbors (k-NN) classifier [10, 11] is a simple and intuitive algorithm used for both classification and regression tasks. Similar to the logistic regression model, the k-NN predicts the class label of a new data point. It works on the principle of considering the class labels of its k-nearest neighbors in the feature space. To predict the class label, the k-NN stores the entire training data information and uses it directly in the prediction phase. A euclidean distance ($d(X_i, X_{new})$) is being calculated between the training data sample (X_i) and the new (test) data sample (X_{new}).

$$d(X_i, X_{new}) = \sqrt{\sum (x_i - x_{i,new})^2} \tag{3.6}$$

where x_i and $x_{i,new}$ represent the corresponding values of the i_{th} feature in X_i and X_{new}, respectively. Calculate the value of k-nearest neighbors by measuring the smallest distance between k-training samples and new data point. Label the class for new data point. The value of k determines the number of nearest neighbors considered while predicting the class. An optimized value of k plays an important role in classification. The smaller or the much larger k-value might affect the peformance of the k-NN classifier, and hence a proper hyperparameter tuning is crucial to achieve good results.

3.1.4 Gaussian Naive Bayes

It is basically a conditional probabilistic machine learning algorithm [12] used in classification. It is based on the Bayes theorem. The use of the word "Naive" implies that given the dependent variable y, the features X are conditionally independent. This model often perform well and can be especially effective when dealing with the high-dimensional datasets. The conditional probability of dependent variable (y) given independent features (X) can be calculated using Bayes theorem.

$$P(y|X) = \frac{(P(X|y) * P(y)}{P(X)} \tag{3.7}$$

where, $P(y|X)$ is the probability of occurrence of class y given the features X. It is known as posterior probability. $P(X|y)$ is the probability of observing the features X given the class y and is referred as likelihood. $P(y)$ is the prior probability of class y. It represents the probability of observing y in the given dataset. $P(X)$ represents the probability of observing the features X in the entire dataset.

In case of continuous independent features and if they are following a Gaussian distribution, the Gaussian Naive Bayes is perfect model to use.

3.1.5 Decision Tree Classifier

The decision tree is a machine learning algorithm [6] used for classification and regression problems. It is a tree-structured model that recursively splits the training data into subsets based on the value of a selected attribute or feature until a stopping criterion is met. The decision tree algorithm creates the tree in a top-down manner and chooses the feature that best separates the data into different classes or values. Once constructed, the decision tree can make predictions on new data by traversing the tree based on the values of the new data's features. Decision trees have advantages such as handling both categorical and continuous data, simplicity and interpretability, and handling missing data. However, they can be prone to overfitting, and the choice of split criterion and stopping criterion can impact the accuracy of the model. Broadly, there are two types into which the various machine learning techniques described above can be classified into.

- **Unsupervised Learning:** The biggest difference between unsupervised learning and supervised learning is absence of data labels in training. Training samples for unsupervised learning have no labels and no definite results for output, the computer needs to learn the similarity between samples by itself and classify the samples. The advantage of unsupervised learning is that there is no need to label, reducing the influence of human subjective factors on the results.

- **Supervised Learning:** Supervised learning [13] needs the labels of the training data. Common supervised learning algorithms include logistic

regression, Naive Bayesian, Support Vector Machine, artificial neural network and random forest. They use the self-learning characteristic of neural networks to transform cracks recognition into crack probability judgement of each sub-block image in the work.

3.1.6 Random Forest Classifier

Random Forest is a popular machine learning algorithm that belongs to the family of ensemble methods. It is a type of supervised learning algorithm that is used for both classification and regression tasks [14]. Random Forest works by constructing multiple decision trees at training time and outputs the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. The individual decision trees are generated using a random selection of features and a random subset of the training data, hence the name "Random Forest". The hyper-parameters in random forest are either used to increase the predictive power of the model or to make the model faster. Let's look at the hyper-parameters of sklearn's built-in random forest function.

- **Increasing the predictive power:** Firstly, there is the n_estimators hyper-parameter, which is just the number of trees the algorithm builds before taking the maximum voting or taking the averages of predictions. In general, a higher number of trees increases the performance and makes the predictions more stable, but it also slows down the computation.

  Another important hyper-parameter is max_features, which is the maximum number of features random forest considers to split a node. Sklearn provides several options, all described in the documentation.

  The last important hyper-parameter is min_sample_leaf. This determines the minimum number of leafs required to split an internal node.

- **Increasing the random forest model's speed:** The n_jobs hyper-parameter tells the engine how many processors it is allowed to use. If it has a value of one, it can only use one processor. A value of "-1" means that there is no limit.

  The random_state hyper-parameter allows the model's output to be replicated. The model will always produce the same results when it has a definite value random_state and if it has been given the same hyper-parameters and the same training data.

  Lastly, there is the oob_score (out-of-bag score) (also called oob_sampling), which is a random forest cross-validation method. In this sampling, about one-third of the data is not used to train the model and can be used to evaluate its performance. These samples are called the out-of-bag samples. It's very similar to the leave-one-out-cross-validation method, but almost no additional computational burden goes along with it.

3.2 Deep Neural Network

A deep neural network (DNN) [15] is a type of artificial neural network (ANN) consists of multiple hidden layers of interconnected nodes (neurons) organized in a sequential manner. The DNNs are a basic building block of deep learning which focuses on the learning from large datasets. These networks are very powerful in handling the complex data structures, non-linear relationships as well as high dimensional data. The main purpose of a neural network is to receive a set of inputs, perform progressively complex calculations on them, and give output to solve real world problems like classification. We restrict ourselves to feed forward neural networks.

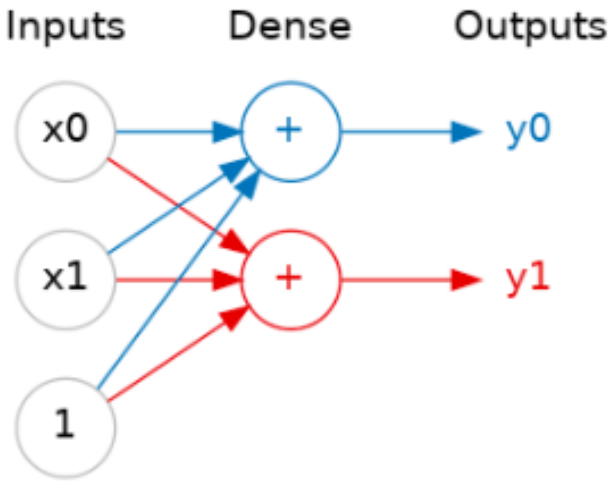


FIGURE 3.1: A typical neural network organized their neurons into layers.

For the DNN model, we have an input, an output, and a flow of sequential data in a deep network. Figure 3.2 represents the Deep Neural Network to understand the complex hidden patterns using dense layers.

Neural networks are widely used in supervised learning and reinforcement learning problems. These networks are based on a set of layers connected to each other. In deep learning, the number of hidden layers, mostly non-linear, can be large; say about 1000 layers. Deep Learning models produce much better results than normal machine learning networks. We mostly use the gradient descent method for optimizing the network and minimising the loss function.

3.2.1 Working of DNN

- **Input Layer:** The first layer of a DNN is the input layer, where the independent features are fed into the network. Each neuron in this layer corresponds to a feature of the input data.

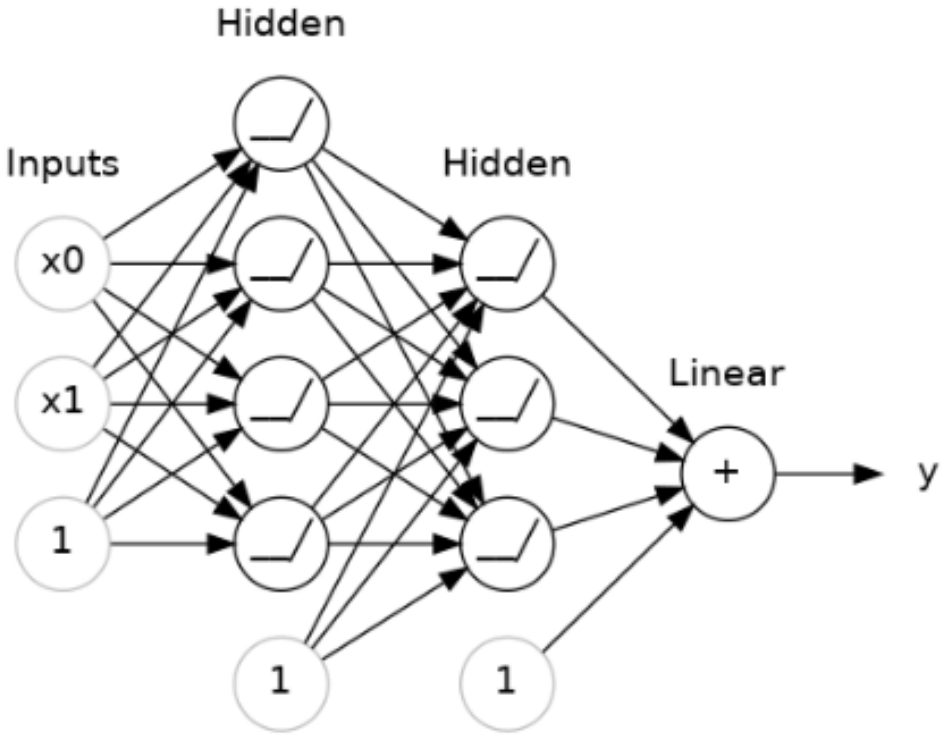


FIGURE 3.2: A deep neural network organized their neurons into multiple layers containing input layer, dense hidden layers, and an output layer. A numerous layers are stacked making a "fully-connected" network to get complex transformations.

- **Hidden Layers:** In between the input and the output layers, there are one (in case of simple model) or more hidden layers. These layers consists of multiple neurons where the output of each neuron is further fed to the input of all neurons in the subsequent layer. This process will continue till the last hidden layer. These hidden layers serve as intermediate layers that learn the hidden patterns, hierarchical representations of the given data.

- **Weights and Biases:** The weights represent the connection between neurons from the different layers. Each connection is associated with a weight that determines the strength of the connection. In addition to the weight, each neuron is also has associated bias allowing the model to learn the correct transformation of the data.

- **Activation function:** Once the weighted sum of each neuron is calculated, an activation function is applied to introduce the non-linearity to the model. There are different types of activation functions such as, ReLU (Rectified Linear Unit), sigmoid, and tanh. These activation functions introduce non-linearity to help the model in understanding the hidden complex patterns in the data.

- **Loss Function:** The performance of the DNN model is evaluated using the loss function. It measures the disparity between the predicted output and the true output for a given input data. In order to increase the accuracy of the model, main goal is to minimize the loss function.
- **Optimization:** A several algorithms have been used to optimize the performance of the model by adjusting the weights to minimize the loss. The optimizer algorithms used in DNN train a network in steps. First, train the model by running it through the network and make prediction. Second, Measure the loss function and in the third step, adjust the weights in a direction that makes the loss smaller than the previous step. Perform these steps over and over again until the loss is as small as it will not decrease further.
- **Batch size and Epochs:** Each iteration has a sample of training data called as batch while a complete round of the training data is called an epoch. Adam is a general purpose optimizer that has an adaptive learning rate making it suitable for most of the problems.

Deep Learning nets are increasingly used for dynamic images apart from static ones and for time series and text analysis. Training the data sets forms an important part of Deep Learning models. In addition, Backpropagation is the main algorithm in training Deep Learning models. Deep Learning deals with training large neural networks with complex input output transformations.

3.3 Libraries used in Machine Learning

The implementation of various machine learning models include several steps. For that matter, one needs to use various libraries [11, 16], such as numpy, pandas, matplotlib, uproot, tensor flow, keras, scikit-learn etc.

- **Numpy [17]:** It is a basic library for numerical computing in Python. Numpy provides support for handling large, multi-dimensional arrays along with the collection of mathematical functions to operate efficiently.
- **Pandas [18]:** It offers data structures such as DataFrame and Series. Pandas can be used for preprocessing, cleaning, aggregation, and for transforming the data.
- **Matplotlib.pyplot [19]:** It is mainly used for data visualization in Python by providing a MATLAB-like interface for generating plots, bar charts, scatter plots, histograms etc.
- **Uproot [20]:** It is a Python library designed to read and write ROOT files commonly used in high energy physics data analysis. ROOT is developed by CERN and uproot helps to access the data stored in ROOT files.

- **TensorFlow [21]:** It is an open-source deep learning library and is most popular and widely used framework for building and training the deep neural networks. Tensor flow provides high-level APIs like Keras, tensorflow.keras, and Estimators which offers user friendly interfaces for building the neural networks. Keras is basically an integrated part of the TensorFlow and is preferred because of its ease of use.

- **Scikit-Learn [22]:** It is a simple and very efficient tool for predictive data analysis. Scikit-learn is an open-source machine learning library that supports supervised and unsupervised learning. It provides numerous tools for data preprocessing, model fitting, model selection, its evaluation and many other utilities.

3.4 Algorithm used for Machine Learning models

- Import numpy, pandas, uproot, and matplotlib.pyplot for mathematical formulas, to read the dataframe, to write and read the ROOT data file, and for data visualization, respectively. Import statements can be written as

```
import numpy as np
import pandas as pd
import uproot
import matplotlib.pyplot as plt
```

  Here, np, pd, and plt are the aliases used for convenience in the python code.

- **Load the data file:** First step is to load the data file and check it thoroughly for any errors in data entries. Access the number of features and dependent variable.

- **Split the data:** The data are split into training and test samples to learn the patterns for the model and to test it afterwards. In our case, the 80% of the data are used to train the model and 20% for the testing purpose. The train_test_split function is available in scikit-learn and is imported as.

```
from sklearn.model_selection import train_test_split
```

  A four different datasets are formed by providing the features and dependent variable X and y, respectively. After using the split function we get.

```
train_x, test_x, train_y, test_y = train_test_split
                                   (
                                       X,
                                       y,
                                       test_size = 0.2,
```

```
                                                    random_state = 40
                                                )
```

- **Data Preprocessing:** Next step is preprocessing the data from a raw form to an organized form for further analysis. For this, StandardScaler has been used. It standardize the features by removing the mean and scaling to unit variance. The scaling is performed independently on each feature by calculating the mean and standard deviation. It is then stored and used on data using transform function. The training and test data for independent features are transformed as.

```
    from sklearn.preprocessing import StandardScaler

    train_x = StandardScaler.fit_transform(train_x)
    test_x = StandardScaler.transform(test_x)
```

- **Machine Learning models:** From scikit-learn (sklearn) import all the machine learning models used for this analysis.

```
    from sklearn.linear_model import LogisticRegression
    from sklearn.svm import SVC
    from sklearn.neighbors import KNeighborsClassifier
    from skelarn.naive_bayes import GaussianNB
    from sklearn.tree import DecisionTreeClassifier
    from sklearn.ensemble import RandomForestClassifier

```

- **DNN:** For the DNN, tensorflow library is used to access the layers and sequential model for the predictive analysis.

```
    import tensorflow as tf
```

  For this thesis, five-layer network with around 160 neurons is used. The Dense layers are defined using 'keras.layers'. The input and output shape is also provided in the first and last layer of the Sequential model, respectively. The example of a model definition is,

```
    model = tf.keras.Sequential([
    tf.keras.layers.Dense(
            units, activation="relu", input_shape=input_shape
            ),
    tf.keras.layers.Dense(units, activation="relu"),
    ...
    ...
    tf.layers.Dense(output_shape, activation="sigmoid"),
    ])

```

Compile the model in the optimizer and loss function.

```
    model.compile(
        loss = "binary_crossentropy",
        optimizer="adam",
        metrics=["accuracy"]
        )
```

In the next step, the model is set to train by providing the batch_size (= 256 rows) of the training data at a time and to perform it for 15 times all the way through the dataset. The number 15 represents epochs. During the model training, the fit method keep on updating the loss values and stores in an object named “History".

```
    History = model.fit(
        train_x,
        y_train,
        validation_data = (test_x, test_y),
        batch_size = batch_size,
        epochs=epochs,
        )
```

In the final step, the loss function and the accuracy score is plotted using the matplotlib.pyplot library.

- **Metrics:** Along with the measurement of the loss function and accuracy score, the confusion matrix is also calculated and plays an important role in deciding the best model. The confusion matrix basically represents the true positives, true negatives, false positives and false negatives predicted by the model.

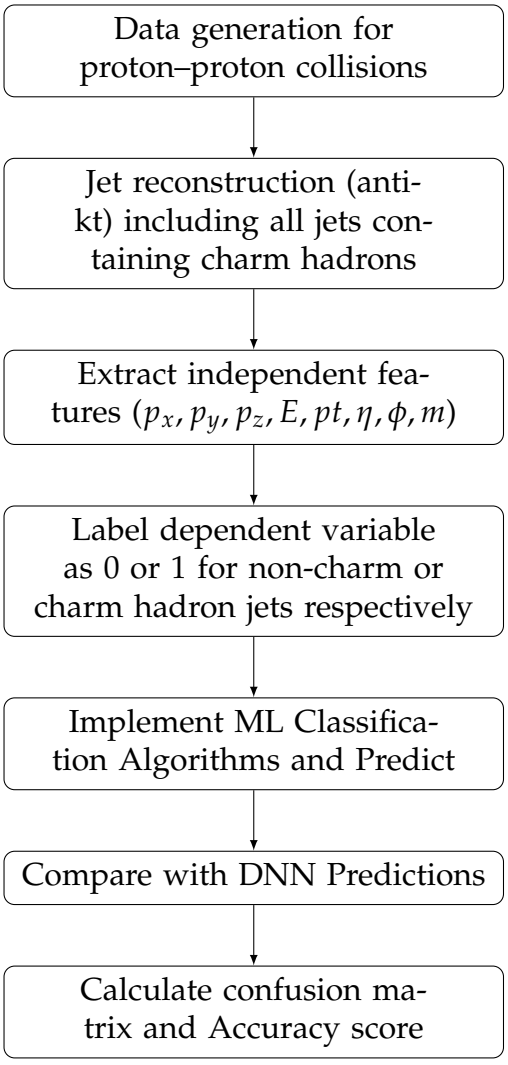
```
    from sklearn.metrics import (
            accuracy_score,
            confusion_matrix
            )
```

The analysis results of the loss function and accuracy score are presented in the next chapter. The comparison of different machine learning models is also discussed.

Chapter 4

Methodology and Analysis Results

To study the charm hadron physics, several steps have been performed. Starting from the data generation, preprocessing it as required, clustering the charm hadrons into jets, extracting features, labelling dependent variable in a binary form, to the implementation of machine learning models. A complete road map for the analysis of charm hadron jet physics is shown in the following flow chart.



4.1 Quality Checks

Around 1000 PYTHIA events have been generated for the charm jet analysis. The pseudorapidity (η), azimuthal angle (ϕ), mupltiplicity of the produced particels in each event are plotted to check the range, consistency, and data completeness. The quality checks also offers a chance to verify the event selection cuts. By conducting these quality checks, one can ensure that the features used in the analysis are reliable and accurate.

- **Pseudorapidity (η):** It is widely used parameter in the high energy physics analysis. Pseudorapidity is a quantity that helps us to understand the angular distribution of particles produced in the high energy collisions. It has the value 0 for particle trajectories that are perpendicular to the beamline, and positive or negative values for those at an angle to the beam. It is defined in terms of the polar angle (θ) of a particle with respect to the beamline direction. The term "beamline" refers to the axis of collision. Mathematically, η can be defined as,

$$\eta = -\ln[\tan(\theta/2)] \tag{4.1}$$

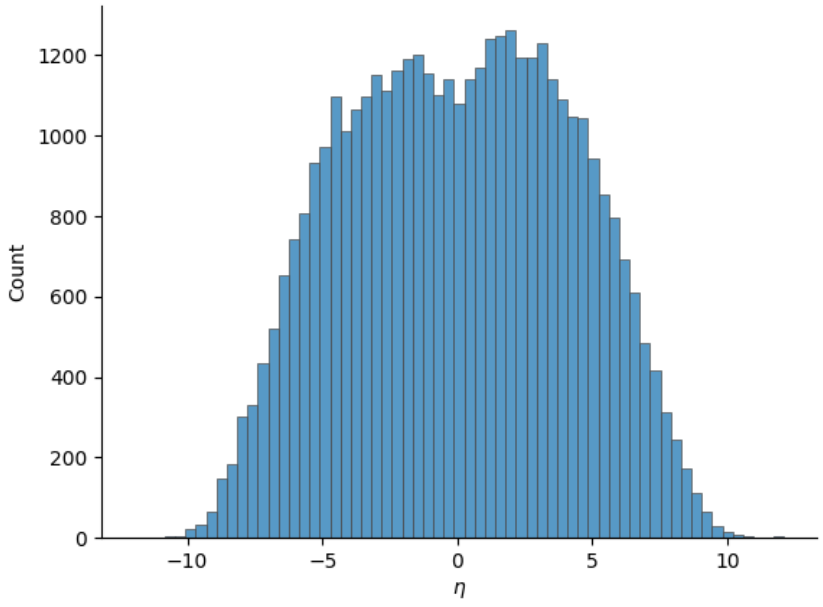


FIGURE 4.1: The pseudorapidity distribution of the produced particles.

Figure 4.1 shows a uniform distribution for the pseudorapidity of the produced particles in a range of -10 to 10. The uniform distribution mean that

the probability of particle produced at any value of eta in a given range (-10 to 10) is constant. The η is one of the coordinate that represents the position of the reconstructed jets making it suitable to use as one of the independent features for the machine learning.

- **Azimuthal Angle (ϕ):** It is an angular measurement in a spherical coordinate system and gives the direction of a particle's momentum in the transverse plane i.e., a plane perpendicular to the beamline direction. The azimuthal angle is measured in radians. ϕ is written as,

$$\phi = \arctan^2(py, px) \tag{4.2}$$

where p_y and p_x are the momentum vectors in the direction of y and x axis, respectively. The value of ϕ lies in the range of (-π, π] or (-180°, 180°] as shown in Figure 4.2. The ϕ is also included in the category of independent features as it locates the position of jet when combines with the η of the jet.

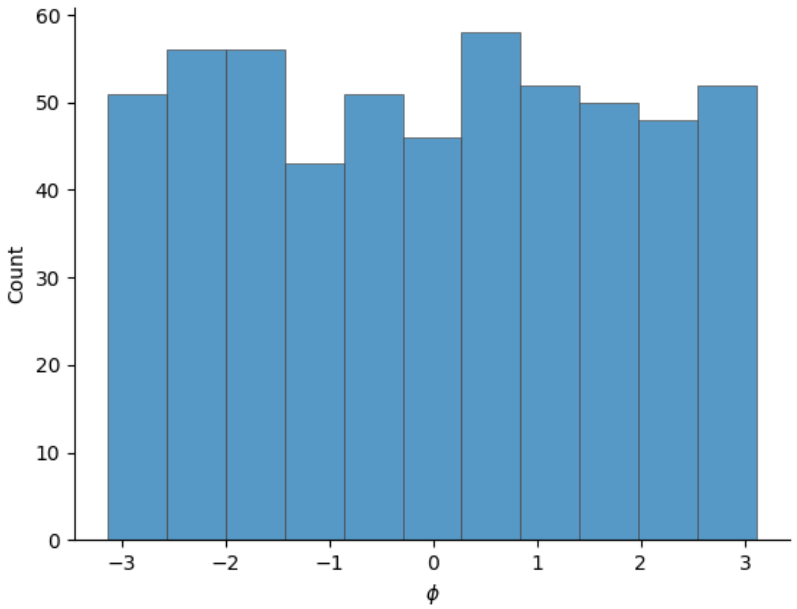


FIGURE 4.2: The azimuthal distribution of the particles ranging from -3.14 to 3.14 radians (-180°, 180°).

- **Charged particle multiplicity:** In high-energy physics, multiplicity refers to the number of particles produced in a collision. It provides information about the density of particle production and used to study the properties of particle interactions. In an event, multiplicity is a count of the total number of particles recorded within a specific range. Multiplicity vary from event

to event and helps to understand the type of collision, such as whether the collision is head-on, or just a peripheral one. Figures 4.3 and 4.4 represent the multiplicity of positively and negatively charged particles in a collision.

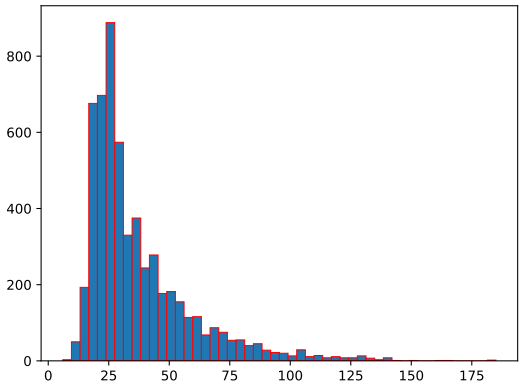


FIGURE 4.3: Multiplicity of negatively charged particles

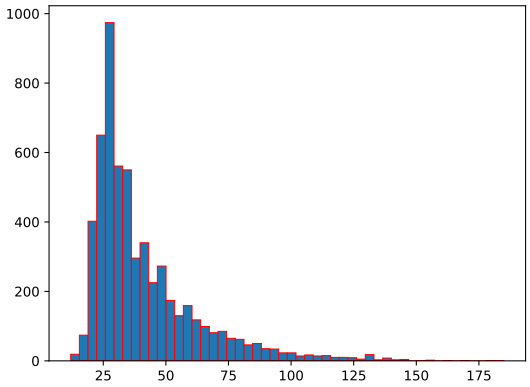


FIGURE 4.4: Multiplicity of positively charged particles

4.2 Results

4.2.1 Charm and Non-Charm Hadron Jets

The dependent variable labelled as 0 or 1 represent non-charm and charm hadron jets present in the data. The bar graph is plotted for both categories and is shown in Figure 4.5 shown by red and blue coloured bars.

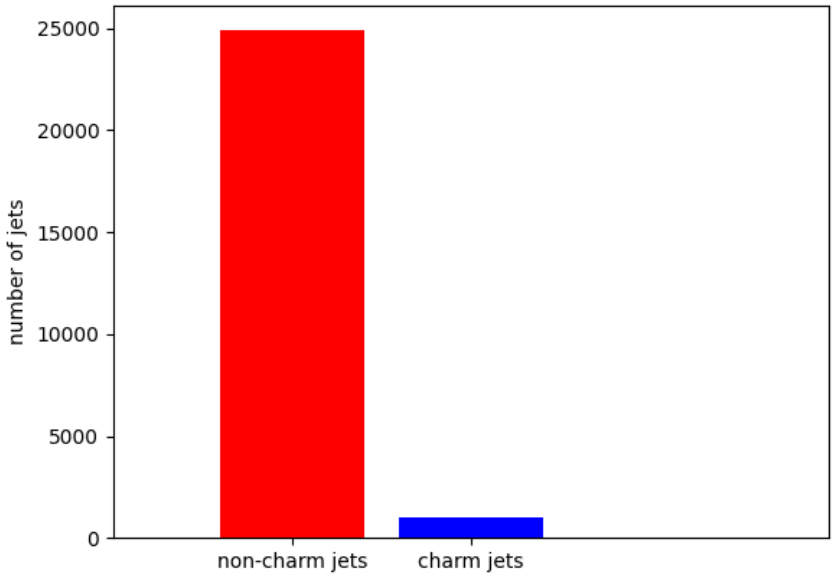


FIGURE 4.5: Number of non-charm and charm hadron jets labelled in the dataset as dependent variable (y).

4.2.2 Variation of Loss Function and Accuracy for the DNN

The loss function used in the DNN is "*binary_crossentropy*" as the output is in the binary form. The optimizer "Adam" is chosen to minimize the loss i.e., to reduce the error between the predicted and the true labels of the charm hadron jets. Figures 4.6 and 4.7 are plotted for the 15 epochs running through the whole dataset with the batch size of 256 referring to the number of rows at a time. From the Figures, it is observed that as the number of epochs are increasing the loss function is showing a decreasing trend for both the training data sample as well as test data sample. Also the loss function for both the training and the test data samples come closer to each other indicating that the DNN is learning from the data and generalizing well to unseen data.
A similar trend of increasing accuracy is observed for both the training and test data samples.

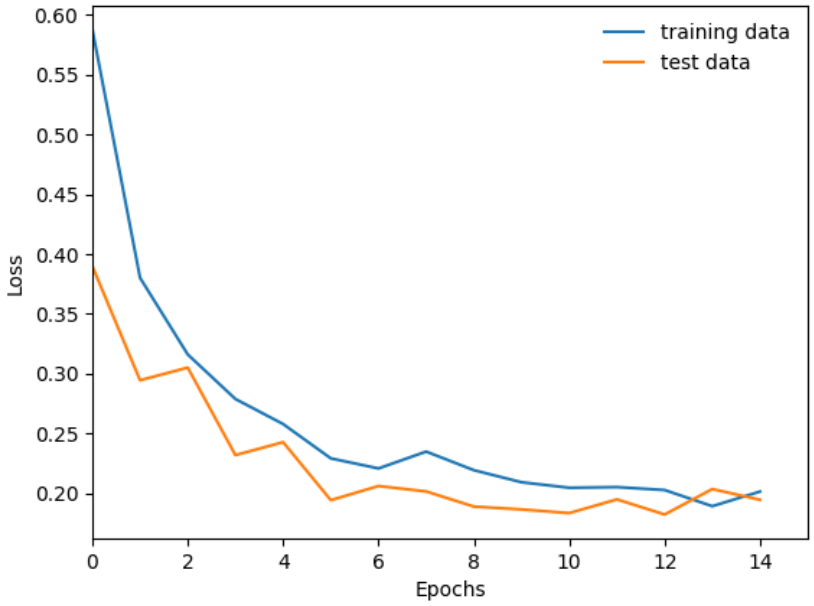


FIGURE 4.6: Variation of the Loss Function for the training and test datasets shown by blue and orange lines, respectively.

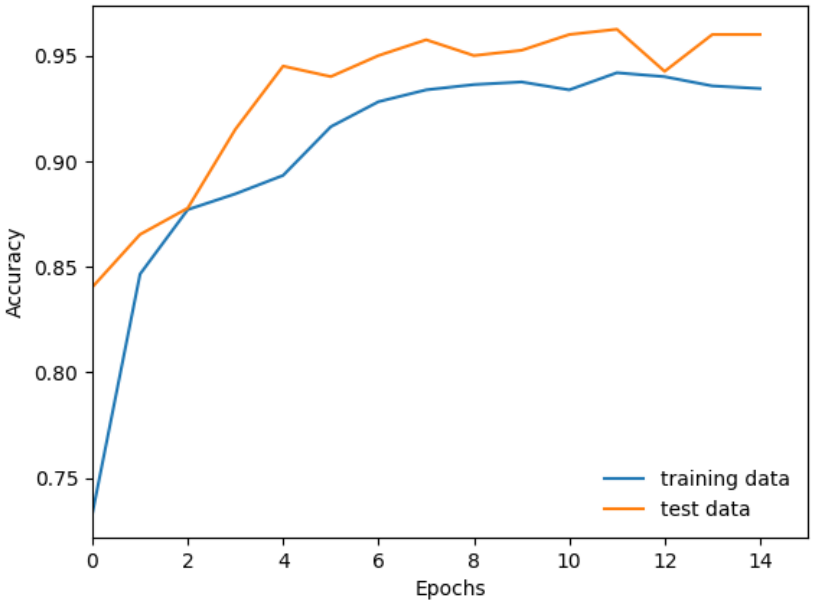


FIGURE 4.7: Variation of the Accuracy score for the training and test datasets shown by blue and orange lines, respectively.

4.2.3 Metrics from the ML models

The machine learning models used for the charm jet classification analysis are tabulated in Table 4.1. The models are trained on the training data sample and then tested on the test data to predict the binary outcome represented by 0 or 1. The label 1 represented the presence of charm hadron jet while label 0 refers to the non-charm hadron jet in a given dataset.

ML Models	**Confusion Matrix**	Accuracy
Logistic Regression	$\begin{bmatrix} 174 & 7 \\ 37 & 181 \end{bmatrix}$	0.8897
KNN Classifier (neighbours=10)	$\begin{bmatrix} 167 & 14 \\ 41 & 177 \end{bmatrix}$	0.8621
SVC (kernel="linear")	$\begin{bmatrix} 172 & 9 \\ 12 & 206 \end{bmatrix}$	0.9473
SVC (kernel="rbf")	$\begin{bmatrix} 172 & 9 \\ 13 & 205 \end{bmatrix}$	0.9448
Gaussian NB	$\begin{bmatrix} 174 & 5 \\ 74 & 146 \end{bmatrix}$	0.80
Decision Tree Classifier (criterion="entropy"	$\begin{bmatrix} 174 & 5 \\ 22 & 198 \end{bmatrix}$	0.9323
Random Forest Classifier (criterion="entropy", n_estimators=10)	$\begin{bmatrix} 170 & 11 \\ 8 & 210 \end{bmatrix}$	0.9523
Deep Neural Network	$\begin{bmatrix} 202 & 9 \\ 5 & 183 \end{bmatrix}$	0.9649

TABLE 4.1: Different machine learning classification models with their confusion matrices and accuracy scores.

From the Table 4.1, it is observed that the different machine learning models exhibit different confusion matrix and accordingly the accuracy score. For every model, a confusion matrix is created which represents total true negatives (00) [upper left], false positives (01) [upper right], false negatives (10) [bottom left], and true positives (11) [bottom right]. The true negatives represent the prediction of label 0 equals to the true label 0. The false negatives belong to the category of predicting 0 but the true value is 1 whereas false positives refers to the category of predicting 1 and the true value for the same is 0. At last, the true positives represent the predictions and true values are same. From the confusion matrix one can easily monitor the performance of each machine learning model and then can further optimize the best model by tuning the hyper-parameters.

Chapter 5

Summary

In this thesis, we generated proton-proton collision events using the Monte Carlo Event Generator (PYTHIA) at centre-of-mass energy of 7 TeV. We have investigated and analyzed various track properties in high-energy physics, including four momenta of the particles (p_x, p_y, p_z, E), pseudorapidity (η), azimuthal angle (ϕ), and particle multiplicity. Our study has shed light on the fundamental aspects of particle collisions and provided valuable insights into the behaviour of particles within the detector. Through rigorous quality checks and data preprocessing, we have ensured the reliability and accuracy of the track properties used in our analysis as the "independent features". By applying range checks, consistency checks, and data completeness assessments, we have identified and addressed any outliers or anomalies that could have affected the integrity of our results.

The distribution analysis of η within the range of -10 to 10 has revealed an interesting and significant observation. The pseudorapidity distribution exhibited a uniform pattern, indicating that particles were produced with nearly equal probability across the entire pseudorapidity range. This finding aligns with our theoretical expectations for certain types of high-energy collisions and enhances our confidence in the validity of the data. Additionally, the ϕ has provided meaningful insights into the orientation of particle momenta with respect to the positive x-axis.

The position of the reconstructed jets have been located by using the η and ϕ as the space coordinates. The $\eta - \phi$ phase space allowed us to find the charm hadron jets in each event and labelled as 0 or 1.

Throughout this thesis, we have leveraged powerful Python libraries, including NumPy, Pandas, Matplotlib.pyplot, and uproot, to handle and process the data efficiently. These libraries have proven to be invaluable tools in data manipulation, visualization, and the extraction of essential insights from the dataset. Other libraries used for the data analysis include TensorFlow, keras, and scikit-learn for the implementation of various machine learning models used to train the data for the predictions on the new datasets.

In conclusion, this thesis has contributed significantly to the understanding of the basic physics concepts. The charm quark physics paving the way for further investigations into particle collisions and enabling advancements in our

knowledge of the fundamental constituents of the universe. The analysis conducted herein, trained the different machine learning classification models for the label predictions. The best results are observed from the Deep Neural Network with 96.49% of accuracy and with best confusion matrix having maximum of true positives and true negatives out of all other machine learning models.

The findings presented in this work are expected to have a meaningful impact on the broader scientific community and inspire new avenues of research in the exciting realm of high-energy particle physics. This research also opened a path to the non-physics community to work on the physics data and predict several physics outcomes using machine learning model as an important tool.

Bibliography

[1] Andreas Hoecker. *Physics at the LHC Run-2 and Beyond*. 2016. arXiv: 1611.07864 [hep-ex].

[2] Gavin P. Salam. "Towards Jetography". In: *Eur. Phys. J. C* 67 (2010), pp. 637–686. DOI: 10.1140/epjc/s10052-010-1314-6. arXiv: 0906.1833 [hep-ph].

[3] Matteo Cacciari, Gavin P. Salam, and Gregory Soyez. "FastJet user manual". In: *The European Physical Journal C* 72.3 (2012). DOI: 10.1140/epjc/s10052-012-1896-2. URL: https://doi.org/10.1140%2Fepjc%2Fs10052-012-1896-2.

[4] Christian Bierlich et al. "A comprehensive guide to the physics and usage of PYTHIA 8.3". In: (Mar. 2022). DOI: 10.21468/SciPostPhysCodeb.8. arXiv: 2203.11601 [hep-ph].

[5] Jim Pivarski et al. *Awkward Array*. Oct. 2018. DOI: 10.5281/zenodo.4341376.

[6] Andreas C. Müller and Sarah Guido. *Introduction to Machine Learning with Python*. O'Reilly Media, 2017.

[7] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[8] Juliana Tolles and William J. Meurer. "Logistic Regression: Relating Patient Characteristics to Outcomes". In: *JAMA* 316.5 (Aug. 2016), pp. 533–534. ISSN: 0098-7484. DOI: 10.1001/jama.2016.7653. eprint: https://jamanetwork.com/journals/jama/articlepdf/2540383/jgm160003.pdf. URL: https://doi.org/10.1001/jama.2016.7653.

[9] Corinna Cortes and Vladimir Vapnik. "Support-Vector Networks". In: *Machine Learning* 20.3 (1995), pp. 273–297.

[10] T. Cover and P. Hart. "Nearest neighbor pattern classification". In: *IEEE Transactions on Information Theory* 13.1 (1967), pp. 21–27. DOI: 10.1109/TIT.1967.1053964.

[11] Sebastian Raschka and Vahid Mirjalili. *Python Machine Learning*. Packt Publishing, 2020.

[12] Gareth James et al. *An Introduction to Statistical Learning*. Springer, 2013.

[13] Sotiris B. Kotsiantis. "Supervised Machine Learning: A Review of Classification Techniques". In: *Informatica* 31.3 (2007), pp. 249–268.

[14] Leo Breiman. "Random Forests". In: *Machine Learning* 45.1 (2001), pp. 5–32.

[15] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[16] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow*. O'Reilly Media, 2017.

[17] Charles R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038%2Fs41586-020-2649-2.

[18] The pandas development team. *pandas-dev/pandas: Pandas*. URL: https://github.com/pandas-dev/pandas.

[19] John D. Hunter. "Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[20] Jim Pivarski et al. *Uproot*. Sept. 2017. DOI: 10.5281/zenodo.4340632.

[21] Martín Abadi et al. *TensorFlow, Large-scale machine learning on heterogeneous systems*. Nov. 2015. DOI: 10.5281/zenodo.4724125.

[22] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.