INSTITUTES FOR ADVANCED COMPUTING AND SOFTWARE DEVELOPMENT

AKRUDI, PUNE

DOCUMENTATION ON

“Identification of Higgs Boson particle by Machine Learning Approach”

PG-DBDA Feb-2020

Submitted By:

Yogesh Mundra (1329)

Manoj Pawar (1334)

Mr. PRASHANT KARHALE
Center Coordinator

Mr. AKSHAY TILEKAR
Project Guide

CERTIFICATE

This is to certify that the Project Entitled

Identification of Higgs Boson particle by Machine Learning Approach

Submitted by

YOGESH MUNDRA	Roll No: 1329
MANOJ PAWAR	Roll No: 1334

is a bonafide student of this institute and the work has been carried out by him/her under the supervision of Mr. AKSHAY TILEKAR and it is approved for the partial fulfillment of the requirement of Post Graduate Diploma in Big Data Analytics.

Mr. AKSHAY TILEKAR Mr. PRASHANT KARHALE

Project Guide Centre Coordinator

ACKNOWLEDGEMENT

It gives us great pleasure in presenting the preliminary project report on 'Identification of Higgs Boson particle by Machine Learning Approach'.

I would like to take this opportunity to thank my internal guide Mr. AKSHAY TILEKAR for giving me all the help and guidance I needed. I am really grateful to them for their kind support. Their valuable suggestions were very helpful.

I am also grateful to Mr. PRASHANT KARHALE, Centre Coordinator, Akurdi, Pune for his indispensable support, suggestions. In the end our special thanks to IACSD for providing various resources and guiding us through entire project completion.

YOGESH MUNDRA
MANOJ PAWAR

CONTENTS

CHAPTER -1

SYNOPSIS

1.1 Problem Statement

To identify the Higgs Boson sub-atomic particle from the jet pull energy of ATLAS Experiment conducted by CERN in Large Hadron Collider (LHC), by performing the Data Analysis on the relevant dataset along with Machine Learning Approach.

1.2 Abstract

In particle physics, Higgs Boson to tau-tau decay signals are notoriously difficult to identify due to the presence of severe background noise generated by other decaying particles. Our approach uses the Machine Learning Algorithms to classify the events as signals and background noise. To capture and study the properties of Elementary Particles after the Proton-Proton collision in LHC is quite difficult. This is due to highly unstable nature of these sub-atomic particles, so eventually it results in coupling of these lighter particles to other heavier particles. The proposed project is based on the identification of the Higgs Boson sub-atomic particle from jet pull energy dataset of the particles' decay, as modeled by the ATLAS Experiment in the Large Hadron Collider (LHC) at CERN. For the performance our system uses the Google Colab tool, viz. free online environment. It provides GPU for processing the data and to give accurate results quickly. It runs entirely on the cloud and no pre-setup is required. Colab enables increase in computing performance by harnessing the power of GPU.

1.3 Goals and Objectives

1.3.1 Goals:

The overall aim of the project is to capture the Higgs Boson sub-atomic particle. To use ensemble Machine Learning Algorithm to enhance Accuracy of the Signal and classifying it from the Noise

CHAPTER 2

INTRODUCTION

The Higgs boson is the Fundamental Particle associated with the Higgs field, a field that gives mass to other fundamental particles such as electrons and quarks. A particle's mass determines how much it resists changing its speed or position when it encounters a force. Not all fundamental particles have mass. The photon, which is the particle of light and carries the electromagnetic force, has no mass at all.

The Higgs boson was proposed in 1964 by Peter Higgs, and four other theorists to explain why certain particles have mass. Scientists confirmed its existence in 2012 through the ATLAS and CMS experiments at the Large Hadron Collider (LHC) at CERN in Switzerland. This discovery led to the 2013 Nobel Prize in Physics being awarded to Higgs and Englert.

Scientists are now studying the characteristic properties of the Higgs boson to determine if it precisely matches the predictions of the Standard Model of particle physics. If the Higgs boson deviates from the model, it may provide clues to new particles that only interact with other Standard Model particles through the Higgs boson and thereby lead to new scientific discoveries.

2.1 Purpose

To identify the Higgs Boson sub-atomic particle, with the capturing of Tau-Tau Particle from the CERN's Dataset of the particles' decay from the Jet-pull energy experimented by ATLAS in the Large Hadron Collider (LHC) at Geneva, Switzerland. It is notoriously difficult to capture and study the properties of Elementary Particles after the Proton-Proton collision in LHC. It is due to highly unstable nature of these particles, so eventually it results in coupling of these lighter particles to other heavier particles.

It'd be helpful for the scientific community and all living beings alike, as this particle (Higgs boson) is responsible for giving the mass to any Elementary Particle of the Standard Model in Particle Physics.

2.2 Scope of the project

2.2.1 Initial functional requirement will be:-

- ❒ Selecting the appropriate algorithms.
- ❒ Determining the appropriate input format to algorithm.
- ❒ Train the model.
- ❒ Test the model.
- ❒ Display the results of the algorithms.

2.2.2 Initial non-functional requirement will be:-

- ❒ Getting the large datasets which can provide developer enough data to train the model.
- ❒ Maintain the minimum variance and bias so the model is successfully work. Avoid
- ❒the under fitting and overfitting.

2.3 Software Development Life Cycle Model

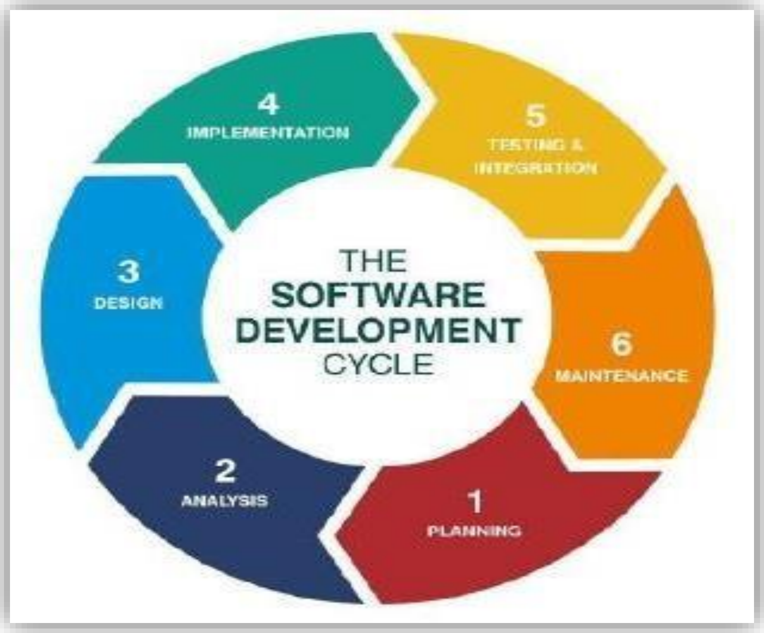


Fig. 2.1 Software Development Life Cycle Model

In order to make this Project we are going to use Classic LIFE CYCLE MODEL .Classic life cycle model is also known as WATERFALL MODEL. The life cycle model demands a Systematic sequential approach to software development that begins at the system level and progress through analysis design coding, testing and maintenance.

2.3.1 The Classic Life Cycle Model

The waterfall model is sequential software development process, in which progress is seen as owing steadily downwards (like a waterfall) through the phases of conception initiation, Analysis, Design (validation), construction. Testing and maintained.

1. System Engineering and Analysis:
Because software is always a part of larger system work. Begins by establishing requirement for all system elements and then allocating some subset of these requirement to the software system Engineering and analysis encompasses the requirement gathering at the system level with a small amount of top level design and analysis.

2. Software requirement Analysis:
The requirement gathering process is intensified and focused specifically on the software requirement for the both system and software are discussed and reviewed with the customer. The customer specifies the entire requirement or the software and the function to be performed by the software.

3. Design:
Software design is actually a multi-step process that focuses on distinct attributes of the program data structure, software architecture, procedural detail and interface of the software that can be assessed or quality before coding begins .Like requirement the design is documented and becomes part of the software.

4. Coding:
The design must be translated into a machine readable form. The coding step performs this task. If design is programmed in a detailed manner, coding can be accomplished mechanically.

5. **Testing:**
Once code has been generated programmed testing begins. The testing process focuses on the internals of the software ensuring that all statement have been tested and on the functional externals hat is conducting tests to uncover the errors and ensure that defined input will produce the results that agree with the required results.

Unit testing:
In computer programming, Unit testing is software Verification and validation method where the programmer gains confidence that individual units of source code are fit to use a unit is the smallest testable part of an application. In procedural programming a unit may be an individual programmed, function, procedure, etc. while in object-oriented programming, the smallest Unit is a class, which may belong to a base/super class abstract class or derived/child class.

Benefits:
The goal of unit testing is to isolate each part of the program and show that the individual parts are correct. A unit test provides a strict written contract that the piece of code must satisfy.

Documentation:
Unit testing provides a sort of living documentation of the system. Developers looking to learn what functionality is provided by a unit and how to use it can look at the tests to gain a basic understanding of the unit API.

Limitation of unit testing: Testing cannot be expected to catch error in the program {It is impossible to evaluate all execution paths for all but the most trivial programs. The same is true for unit testing.
Additionally, by unit testing only types the functionality if the units themselves.

6. Maintenance:
Software will undoubtedly undergo change after it is delivered to the customer .Change will occur because errors have been encountered be- cause the software must be able adopted to accommodate changes in its external environment because the customer requires functional or performance enhancement enhancements. The classic life cycle is the oldest and most widely used paradigm or software engineering.

2.4 CRISP-DM Method Life Cycle

CRISP-DM methodology that stands for Cross Industry Standard Process for Data Mining, is a cycle that describes commonly used approaches that data mining experts use to tackle problems in traditional BI data mining. It is still being used in traditional BI data mining teams. Take a look at the following illustration. It shows the major stages of the cycle as described by the CRISPDM methodology and how they are interrelated.

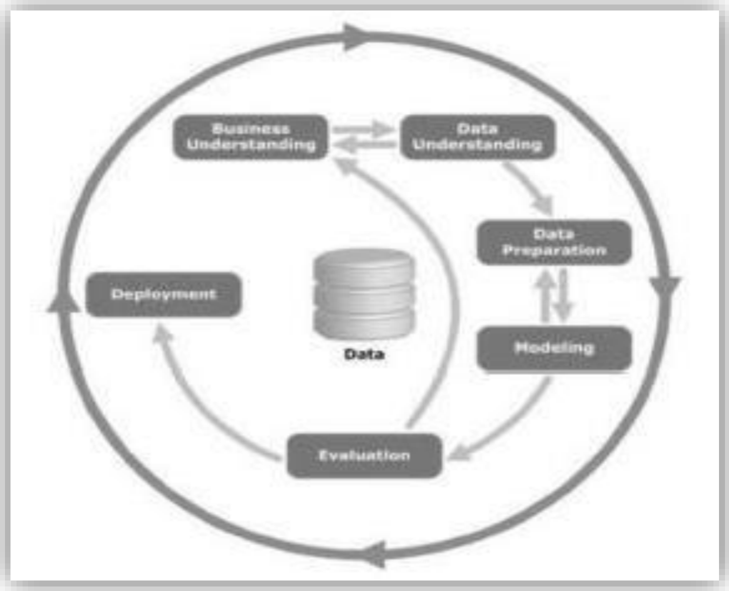


Fig. 2.2: Life Cycle of Data Mining

Let us now learn a little more on each of the stages involved in the CRISP-DM life cycle-

- **Business Understanding:** This initial phase focuses on understanding the project objectives and requirements from a business perspective, and then converting this knowledge into a data mining problem definition. A preliminary plan is designed to achieve the objectives. A decision model, especially one built using the Decision Model and Notation standard can be used.

- **Data Understanding:** The data understanding phase starts with an initial data collection and proceeds with activities in order to get familiar with the data, to identify data quality problems, to discover first insights into the data, or to detect interesting subsets to form hypotheses for hidden information.

- **Data Preparation:** The data preparation phase covers all activities to construct the final dataset (data that will be fed into the modeling tool(s)) from the initial raw data. Data preparation tasks are likely to be performed multiple times, and not in any prescribed order.

Tasks include table, record, and attribute selection as well as transformation and cleaning of data for modeling tools.

- **Modeling:** In this phase, various modeling techniques are selected and applied and their parameters are calibrated to optimal values. Typically, there are several techniques for the same data mining problem type. Some techniques have specific requirements on the form of data. Therefore, it is often required to step back to the data preparation phase.

- **Evaluation:** At this stage in the project, you have built a model (or models) that appears to have high quality, from a data analysis perspective. Before proceeding to final deployment of the model, it is important to evaluate the model thoroughly and review the steps executed to construct the model, to be certain it properly achieves the business objectives. A key objective is to determine if there is some important business issue that has not been sufficiently considered. At the end of this phase, a decision on the use of the data mining results should be reached.

- **Deployment:** Creation of the model is generally not the end of the project. Even if the purpose of the model is to increase knowledge of the data, the knowledge gained will need to be organized and presented in away that is useful to the customer. Depending on the requirements, the deployment phase can be as simple as generating a report or as complex as implementing a repeatable data scoring (e.g. segment allocation) or data mining process.

2.5 Overview of Document

Extracting Data: Data is gathered from the data website, Kaggle; The dataset has been built from official ATLAS full-detector simulation, with "Higgs to tautau" events mixed with different backgrounds.

Objective: Main goal is to classify the events of the dataset inyo signal and background Noise also find out the best fitted model with the help of Accuracy as well as F1 Score.

Imports: This section describes way to connect external analysis requirements to python.

- **Cleaning and preparation:** Section describes procedure to cleaning the data to make it analysis ready.
- **Analysis:** This is most important section of project describing all analysis done and its visualization.
- **Conclusion:** Conclusion concludes the project.

CHAPTER – 3
OVERALL DESCRIPTION

3.1 Dataset Description

3.1.1 Dataset from the ATLAS Higgs Boson Machine Learning Challenge:

The dataset has been built from official ATLAS full-detector simulation, with "Higgs to tautau" events mixed with different backgrounds. The simulator has two parts. In the first, random proton-proton collisions are simulated based on the knowledge that have been accumulated on particle physics. It reproduces the random microscopic explosions resulting from the proton-proton collisions. In the second part, the resulting particles are tracked through a virtual model of the detector. The process yields simulated events with properties that mimic the statistical properties of the real events with additional information on what has happened during the collision, before particles are measured in the detector.

- EventId - An unique integer identifier of the event.
- DER_mass_MMC - The estimated mass mHmH of the Higgs boson candidate, obtained through a probabilistic phase space integration.
- DER_mass_transverse_met_lep - The transverse mass between the missing transverse energy and the lepton.
- DER_mass_vis - The invariant mass of the hadronic tau and the lepton.
- DER_pt_h - The modulus of the vector sum of the transverse momentum of the hadronic tau, the lepton and the missing transverse energy vector.
- DER_deltaeta_jet_jet - The absolute value of the pseudorapidity separation between the two jets (undefined if PRI_jet_num ≤≤ 1).
- DER_mass_jet_jet - The invariant mass of the two jets (undefined if PRI_jet_num ≤≤ 1).
- DER_prodeta_jet_jet - The product of the pseudorapidities of the two jets (undefined if PRI_jet_num ≤≤ 1).
- DER_deltar_tau_lep - The R separation between the hadronic tau and the lepton.

- DER_pt_tot - The modulus of the vector sum of the missing transverse momenta and the transverse momenta of the hadronic tau, the lepton, the leading jet (if PRI_jet_num ≥≥) and the subleading jet (if PRI jet num = 2) (but not of any additional jets).
- DER_sum_pt - The sum of the moduli of the transverse momenta of the hadronic tau, the lepton, the leading jet (if PRI jet num ≥≥ 1) and the subleading jet (if PRI jet num = 2) and the other jets (if PRI jet num = 3).
- DER_pt_ratio_lep_tau - The ratio of the transverse momenta of the lepton and the hadronic tau.
- DER_met_phi_centrality- The centrality of the azimuthal angle of the missing transverse energy vector w.r.t. the hadronic tau and the lepton.
- DER_lep_eta_centrality - The centrality of the pseudorapidity of the lepton w.r.t. the two jets (undefined if PRI_jet_num ≤≤ 1).
- PRI_tau_pt - The transverse momentum √px2+py2 of the hadronic tau.
- PRI_tau_eta - The pseudorapidity η of the hadronic tau.PRI_tau_phi-
- PRI_lep_pt – The azimuth angle ϕ of the hadronic tau.
- PRI_lep_eta - The transverse momentum √px2+py2 of the lepton (electron or muon).
- PRI_lep_phi - The azimuth angle ϕ of the lepton.
- PRI_met - The missing transverse energy E→missT
- PRI_met_phi - The azimuth angle ϕϕ of the mssing transverse energy
- PRI_met_sumet - The total transverse energy in the detector.
- PRI_jet_num - The number of jets (integer with value of 0, 1, 2 or 3; possible larger values have been capped at 3).
- PRI_jet_leading_pt - The transverse momentum √px2+py2 of the leading jet, that is the jet with largest transverse momentum (undefined if PRI_jet_num = 0).
- PRI_jet_leading_eta - The pseudorapidity η of the leading jet (undefined if PRI jet num = 0).
- PRI_jet_leading_phi - The azimuth angle ϕϕ of the leading jet (undefined if PRI jet num = 0).

- PRI_jet_subleading_pt - The transverse momentum √px2+py2 of the leading jet, that is, the jet with second largest transverse momentum (undefined if PRI_jet_num ≤≤ 1).
- PRI_jet_subleading_eta - The pseudorapidity η of the subleading jet (undefined if PRI_jet_num ≤≤ 1).
- PRI_jet_subleading_phi - The azimuth angle ϕ of the subleading jet (undefined if PRI_jet_num ≤≤ 1).
- PRI_jet_all_pt - The scalar sum of the transverse momentum of all the jets of the events.
- Weight - The event weight wi
- Label - The event label (string) yi ∈ {s,b}(s for signal, b for background).

3.2 Imports

- **Matplotlib:** Matplotlib is a collection of command style functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure. It is used to show visualizations of analysis.

- **Pandas:** Pandas is an open-source, BSD-licensed Python library providing high-performance, and easy-to-use data structures and data analysis tools for the Python programming language. Pandas is used to perform operation on data.

- **Numpy:** Numpy is used to for mathematical operations. This package provides easy use of mathematical function.

- **Seaborn:** Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

- **Sklearn:** Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including SVM.

CHAPTER – 4
REQUIREMENT SPECIFICATION

4.1 External Requirement Specification

4.1.1 Hardware Interfaces

It deals with the hardware tools which gives the better understanding of the project.

- Processor: Intel core i5 processor with 1.80 GHz processor or more.
- RAM: 8 or 12 GB
- Hard disk: Minimum 500 GB
- LAN: High speed upto 100 Mbps

4.1.2 Software Requirements

- Operating System: Windows 10
- Back end: Python
- Anaconda Navigator
- Jupyter Notebook
- Google Colab tool

CHAPTER – 5
DATA PRE-PROCESSING & ANALYSIS

5.1 Data Pre-processing:

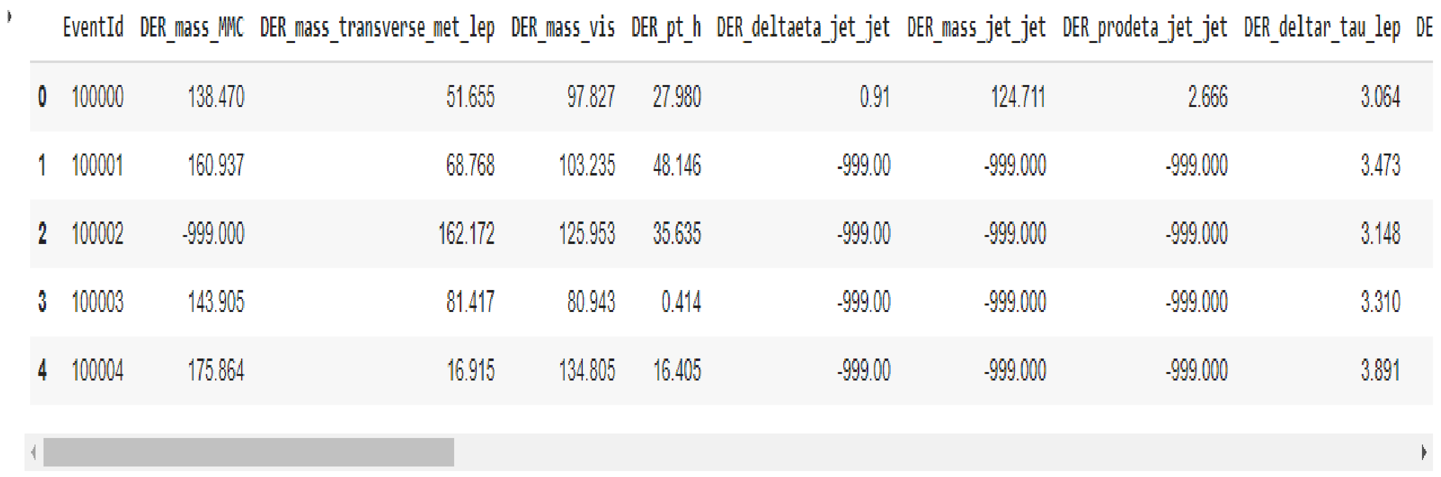
Data preprocessing includes merging and manipulating data from dataset, performing data visualization, imputing new variables, selecting features using principle component analysis (PCA), and splitting data into train and test datasets.

Unrealistic and outlier data will be removed to develop a representative and more general model. New variables can be created by extracting useful information from current data or external resources.

Data visualization aims to create various plots so that we can easily tell the distribution of a feature or the relationships between features. Plots involved include histogram, boxplot, correlation plot, etc. Data visualization is a convenient way to make a first judgement on the importance of variables in prediction. Besides, model assumptions can be easily checked using graphs.

It is essential to check multicollinearity before developing the model. Some machine learning models, random forest and gradient boosting for example, can handle multicollinearity well. However, it will be a problem in prediction using logistic regression and Naïve Bayes.

5.1.1 Screenshot before Cleaning

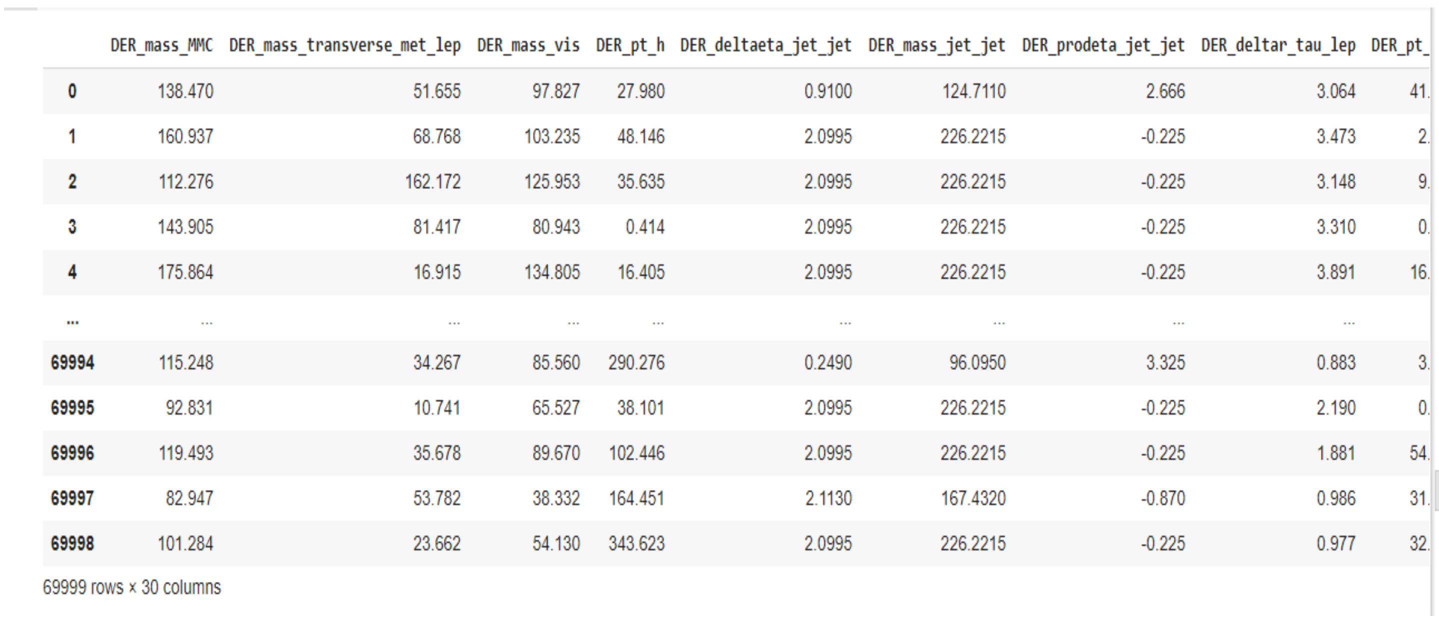


	EventId	DER_mass_MMC	DER_mass_transverse_met_lep	DER_mass_vis	DER_pt_h	DER_deltaeta_jet_jet	DER_mass_jet_jet	DER_prodeta_jet_jet	DER_deltar_tau_lep	DE
0	100000	138.470	51.655	97.827	27.980	0.91	124.711	2.666	3.064	
1	100001	160.937	68.768	103.235	48.146	-999.00	-999.000	-999.000	3.473	
2	100002	-999.000	162.172	125.953	35.635	-999.00	-999.000	-999.000	3.148	
3	100003	143.905	81.417	80.943	0.414	-999.00	-999.000	-999.000	3.310	
4	100004	175.864	16.915	134.805	16.405	-999.00	-999.000	-999.000	3.891	

Fig. 5.1.1 Data Before Cleaning

In this Figure, the data has some negative values like -999 because of that the accuracy can get wrong model.

5.1.2 Screenshot after Cleaning



	DER_mass_MMC	DER_mass_transverse_met_lep	DER_mass_vis	DER_pt_h	DER_deltaeta_jet_jet	DER_mass_jet_jet	DER_prodeta_jet_jet	DER_deltar_tau_lep	DER_pt_
0	138.470	51.655	97.827	27.980	0.9100	124.7110	2.666	3.064	41.
1	160.937	68.768	103.235	48.146	2.0995	226.2215	-0.225	3.473	2.
2	112.276	162.172	125.953	35.635	2.0995	226.2215	-0.225	3.148	9.
3	143.905	81.417	80.943	0.414	2.0995	226.2215	-0.225	3.310	0.
4	175.864	16.915	134.805	16.405	2.0995	226.2215	-0.225	3.891	16.
...	...	...	...	...	...	...	...	...	
69994	115.248	34.267	85.560	290.276	0.2490	96.0950	3.325	0.883	3.
69995	92.831	10.741	65.527	38.101	2.0995	226.2215	-0.225	2.190	0.
69996	119.493	35.678	89.670	102.446	2.0995	226.2215	-0.225	1.881	54.
69997	82.947	53.782	38.332	164.451	2.1130	167.4320	-0.870	0.986	31.
69998	101.284	23.662	54.130	343.623	2.0995	226.2215	-0.225	0.977	32.

69999 rows × 30 columns

Fig. 5.1.2: Data after Cleaning

In this Figure, the data has all Negative values gets removed and replaced with median using Simple Imputer method also data gets under sampled and because of that the accuracy getting is better.

5.2 Finding Correlation

Relationship can be an important tool for feature engineering in building machine learning models. (Describe a possible future event) which are uncorrelated with the goal (number or thing that changes) are probably good candidates to trim from the model. Also, if two (describe a possible future event) are strongly related to each other, then we only need to use one of them. Taking these steps means that the resulting model will be simpler, and simpler models are easier to understand/explain. There are many measures for relationship, but by far the most widely used one is Pearson's Product-Moment coefficient, or Pearson's r. Given a collection of paired (x, y) values, Pearson's coefficient produces a value between -1 and +1 to put into numbers the strength of dependence between the (numbers that change/things that change) x and y. A value of +1 means that all the (x, y) points lie exactly on a line with positive slope, and inversely, a value of -1 means that all of the points lie exactly on a line with negative slope. A Pearson's coefficient of 0 means that there is no relationship between the two (numbers that change/things that change).
To see this visually, we can look at plots of our possible data, and the Pearson's coefficient figured Out/calculated from them.

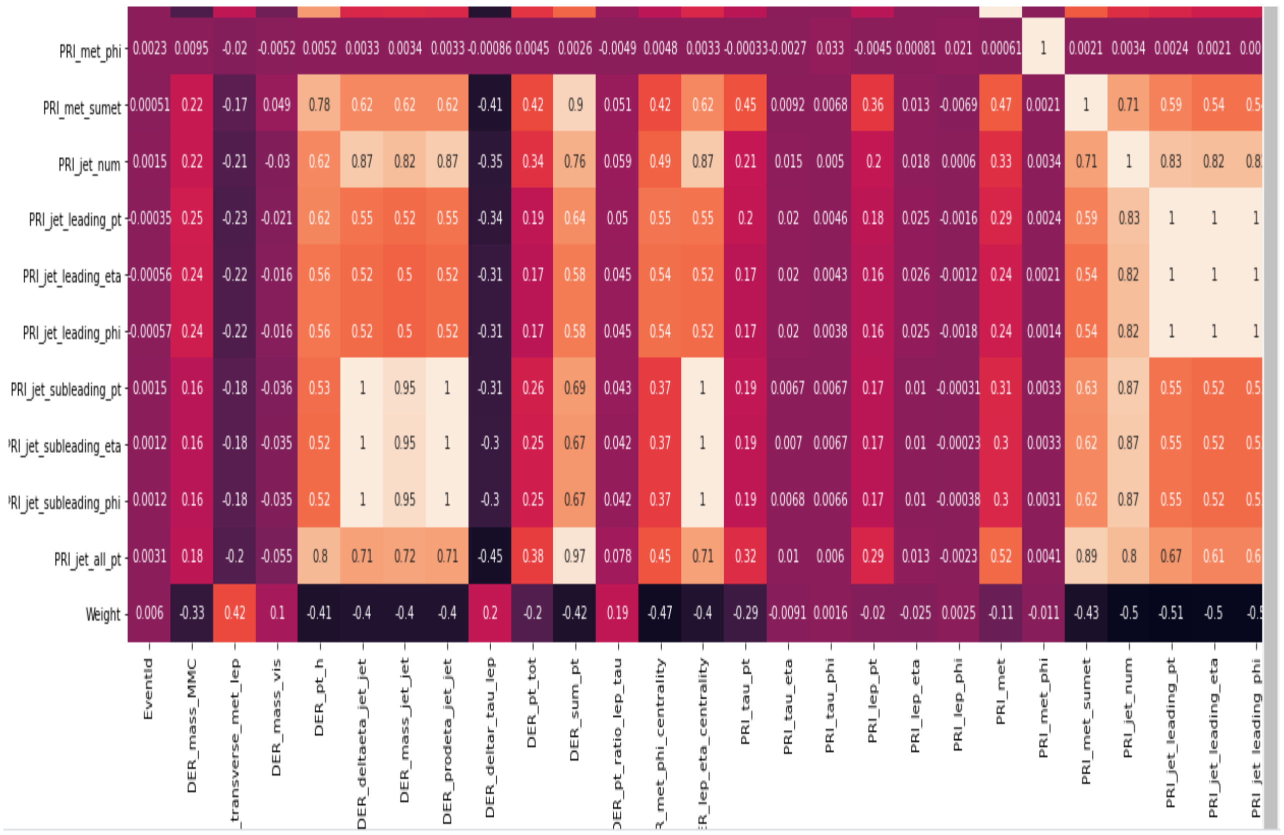
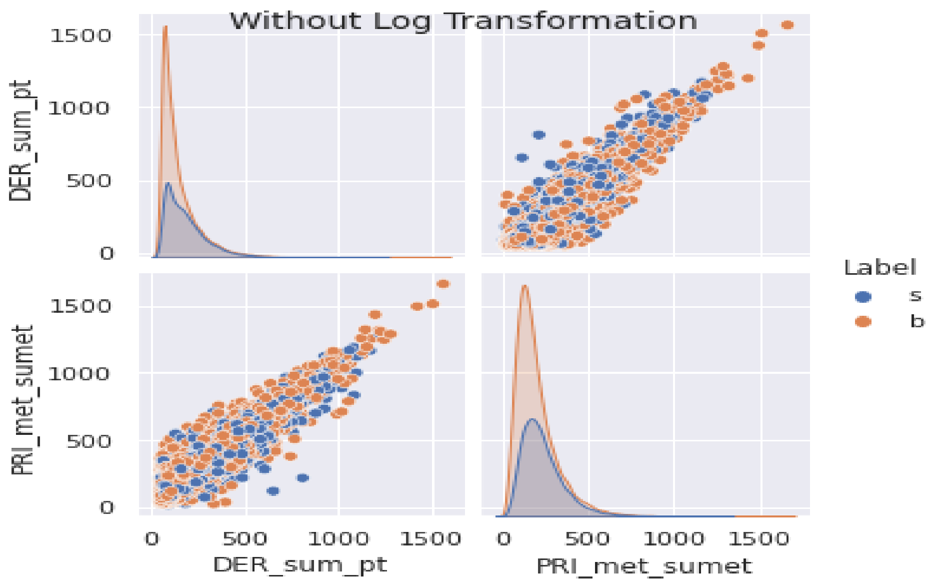


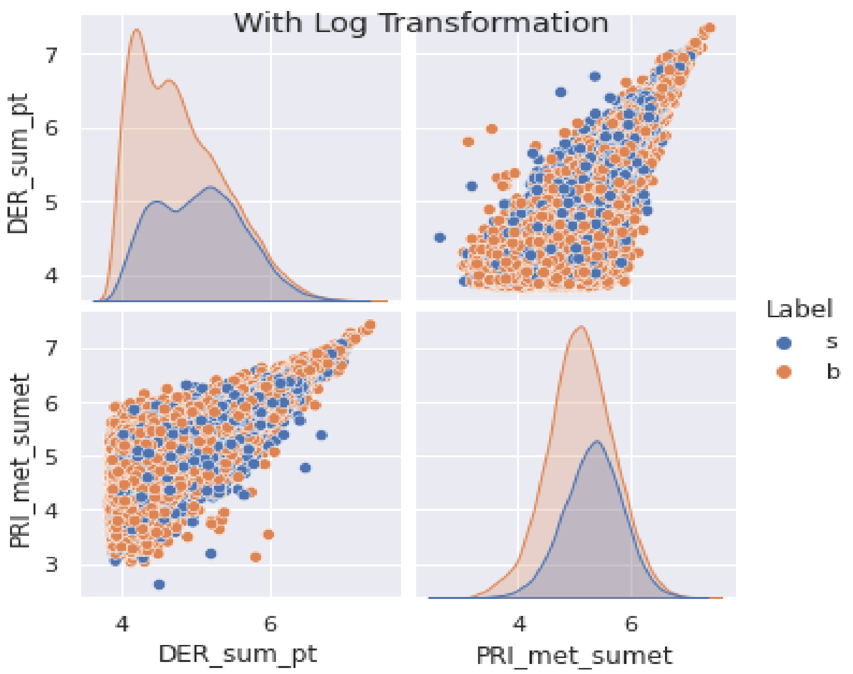
Fig. 5.2.1: correlation heatmap graph

DER_lep_eta_centrality is highly correlated with DER_prodeta_jet_jet (ρ = 0.99999) Rejected
DER_mass_jet_jet is highly correlated with DER_deltaeta_jet_jet (ρ = 0.94604) Rejected
DER_prodeta_jet_jet is highly correlated with DER_mass_jet_jet (ρ = 0.94444) Rejected
PRI_jet_all_pt is highly correlated with DER_sum_pt (ρ = 0.96563) Rejected
PRI_jet_leading_eta is highly correlated with PRI_jet_leading_pt (ρ = 0.9961) Rejected
PRI_jet_leading_phi is highly correlated with PRI_jet_leading_eta (ρ = 0.99999) Rejected
PRI_jet_num has 99913 / 40.0% zeros Zeros
PRI_jet_subleading_eta is highly correlated with PRI_jet_subleading_pt (ρ = 0.99935) Rejected
PRI_jet_subleading_phi is highly correlated with PRI_jet_subleading_eta (ρ = 0.99999) Rejected
PRI_jet_subleading_pt is highly correlated with DER_lep_eta_centrality (ρ = 0.99935) Rejected
PRI_met_sumet is highly correlated with DER_sum_pt (ρ = 0.90448) Rejected

Features DER_deltaeta_jet_jet, DER_mass_jet_jet, DER_prodeta_jet_jet and PRI_jet_subleadi ng_pt, PRI_jet_subleading_eta, PRI_jet_subleading_phi, are the most correlated with target variable

5.3 Log Transformation : The Log Transform is one of the most popular Transformation techniques out there. It is primarily used to convert a skewed distribution to a normal distribution/less-skewed distribution. If the original data follows a log-normal distribution or approximately so, then the log-transformed data follows a normal or near normal distribution.

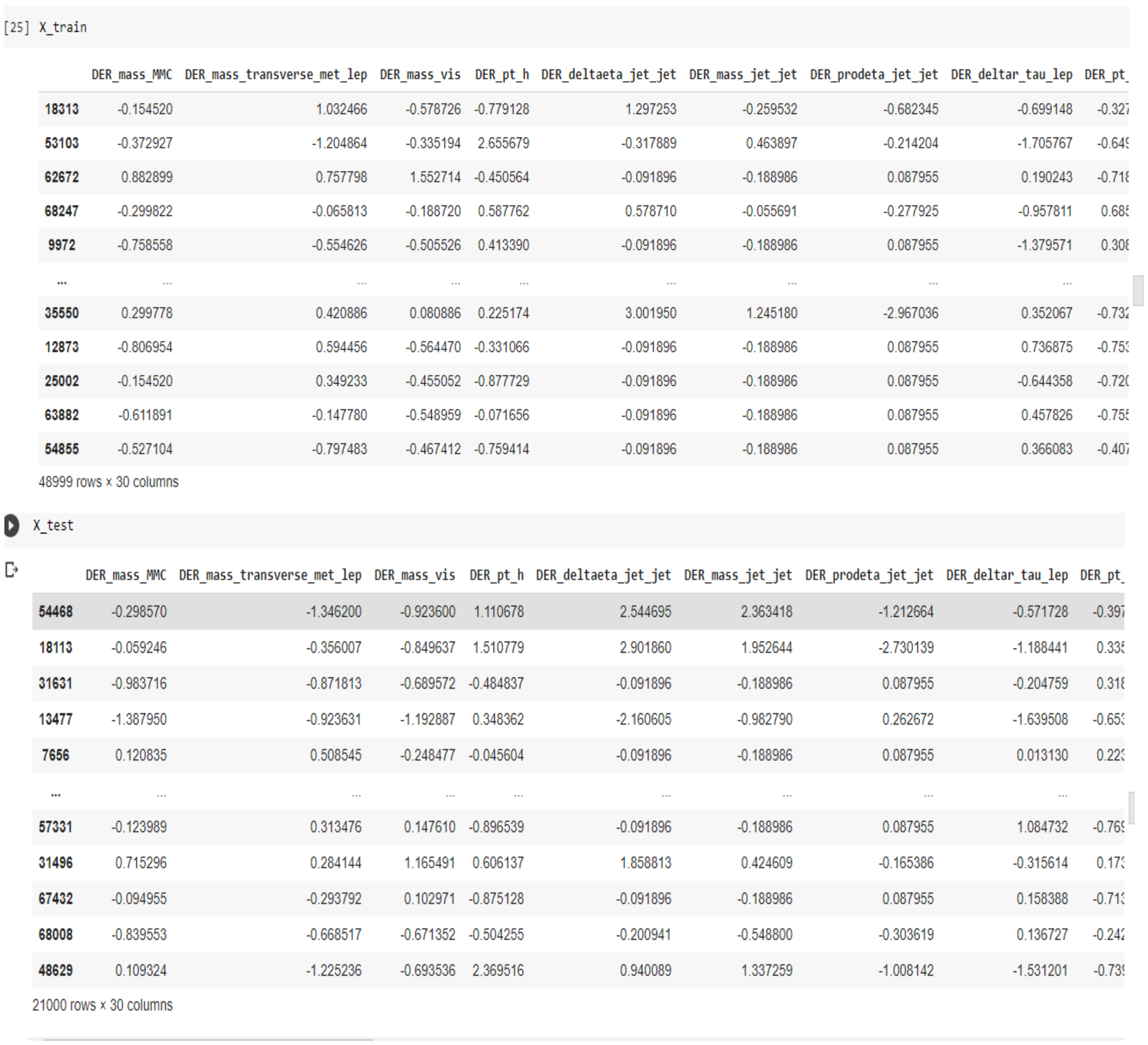




CHAPTER – 6

MODEL EVALUATION

6.1 Training and Testing Data



```
[25] X_train

       DER_mass_MMC DER_mass_transverse_met_lep DER_mass_vis DER_pt_h DER_deltaeta_jet_jet DER_mass_jet_jet DER_prodeta_jet_jet DER_deltar_tau_lep DER_pt_
18313     -0.154520                    1.032466    -0.578726 -0.779128             1.297253        -0.259532           -0.682345          -0.699148  -0.327
53103     -0.372927                   -1.204864    -0.335194  2.655679            -0.317889         0.463897           -0.214204          -1.705767  -0.649
62672      0.882899                    0.757798     1.552714 -0.450564            -0.091896        -0.188986            0.087955           0.190243  -0.718
68247     -0.299822                   -0.065813    -0.188720  0.587762             0.578710        -0.055691           -0.277925          -0.957811   0.685
9972      -0.758558                   -0.554626    -0.505526  0.413390            -0.091896        -0.188986            0.087955          -1.379571   0.308
...             ...                         ...          ...       ...                  ...              ...                 ...                ...
35550      0.299778                    0.420886     0.080886  0.225174             3.001950         1.245180           -2.967036           0.352067  -0.732
12873     -0.806954                    0.594456    -0.564470 -0.331066            -0.091896        -0.188986            0.087955           0.736875  -0.753
25002     -0.154520                    0.349233    -0.455052 -0.877729            -0.091896        -0.188986            0.087955          -0.644358  -0.720
63882     -0.611891                   -0.147780    -0.548959 -0.071656            -0.091896        -0.188986            0.087955           0.457826  -0.755
54855     -0.527104                   -0.797483    -0.467412 -0.759414            -0.091896        -0.188986            0.087955           0.366083  -0.407
48999 rows × 30 columns

X_test

       DER_mass_MMC DER_mass_transverse_met_lep DER_mass_vis DER_pt_h DER_deltaeta_jet_jet DER_mass_jet_jet DER_prodeta_jet_jet DER_deltar_tau_lep DER_pt_
54468     -0.298570                   -1.346200    -0.923600  1.110678             2.544695         2.363418           -1.212664          -0.571728  -0.397
18113     -0.059246                   -0.356007    -0.849637  1.510779             2.901860         1.952644           -2.730139          -1.188441   0.335
31631     -0.983716                   -0.871813    -0.689572 -0.484837            -0.091896        -0.188986            0.087955          -0.204759   0.318
13477     -1.387950                   -0.923631    -1.192887  0.348362            -2.160605        -0.982790            0.262672          -1.639508  -0.653
7656       0.120835                    0.508545    -0.248477 -0.045604            -0.091896        -0.188986            0.087955           0.013130   0.223
...             ...                         ...          ...       ...                  ...              ...                 ...                ...
57331     -0.123989                    0.313476     0.147610 -0.896539            -0.091896        -0.188986            0.087955           1.084732  -0.769
31496      0.715296                    0.284144     1.165491  0.606137             1.858813         0.424609           -0.165386          -0.315614   0.173
67432     -0.094955                   -0.293792     0.102971 -0.875128            -0.091896        -0.188986            0.087955           0.158388  -0.713
68008     -0.839553                   -0.668517    -0.671352 -0.504255            -0.200941        -0.548800           -0.303619           0.136727  -0.242
48629      0.109324                   -1.225236    -0.693536  2.369516             0.940089         1.337259           -1.008142          -1.531201  -0.739
21000 rows × 30 columns
```

Figure 6.1.1: Training and Testing Data

In this Figure 6.1, training phase consists of 70% data and testing phase consists of 30% data.

• **Training Process of Regression Model**

Regression using machine learning algorithm has two phases:

1. Training phase: In this phase the machine learning algorithm is trained using data set comprised of the image and their corresponding labels.
2. Prediction phase: In this phase, the trained model is utilized to predict the labels of unseen image.

The Training phase for classification problems has 2 main steps:

1. Feature Extraction: In this phase, domain knowledge is utilized to extract feature of machine learning algorithm.
2. Model Training: In this phase, clean dataset is utilized to compose of required feature and corresponding label to train the machine learning model.

In the prediction phase, same feature extraction process is applied to the new records and we pass the feature to train the machine learning algorithm to predict the label.

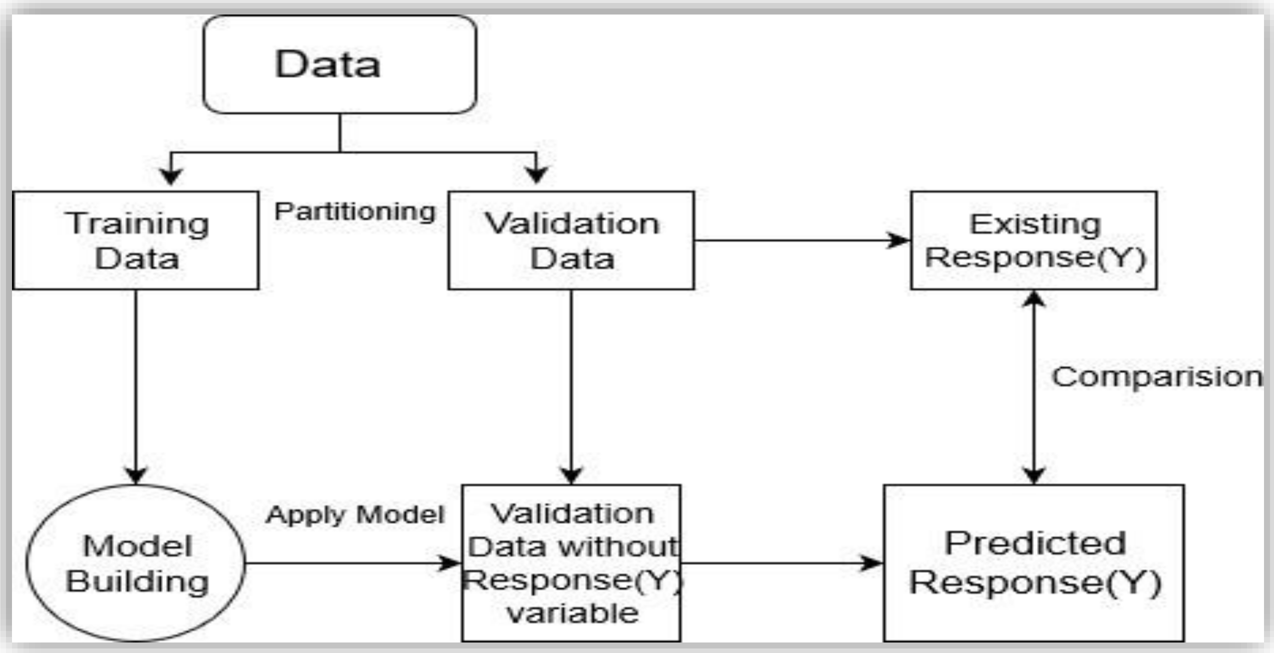


Fig.6.1.2 Machine Learning phase

6.2 Model Fitting

6.2.1 K-Nearest Neighbors Model (KNN): The algorithm looks for observations in our training data that are similar or “near” the record to be classified in the predictor space**.** In k-nearest neighbors method, the classifier identifies k observations in the training dataset that are similar to a new record that we wish to classify.

- Step 1 − For implementing any algorithm, we need dataset. So during the first step of KNN, we must load the training as well as test data.

- Step 2 − Next, we need to choose the value of K i.e. the nearest data points. K can be any integer.

- Step 3 − For each point in the test data do the following −

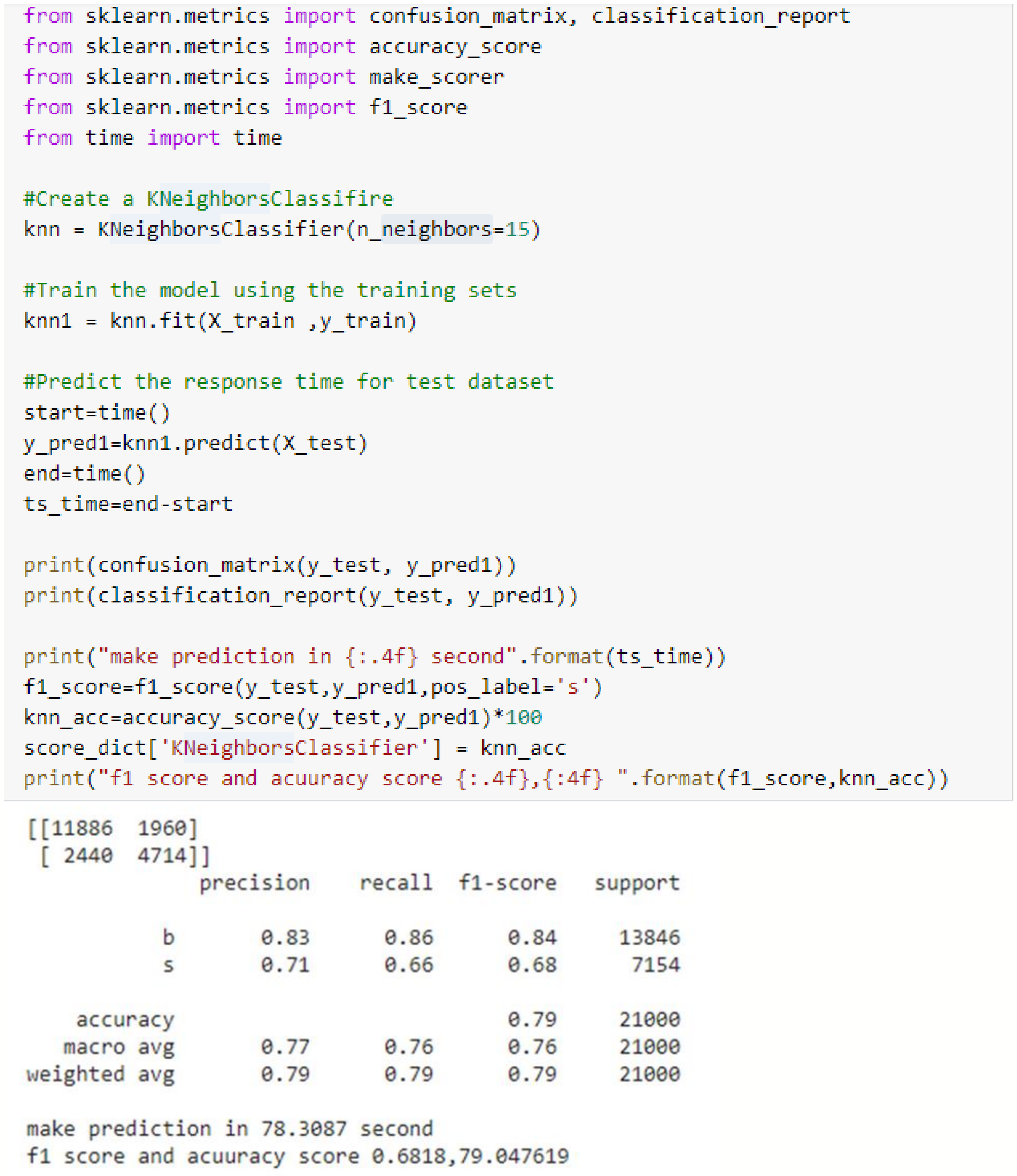
  3.1 − Calculate the distance between test data and each row of training data with the help of any of the method namely: Euclidean, Manhattan or Hamming distance. The most commonly used method to calculate distance is Euclidean.

  3.2 − Now, based on the distance value, sort them in ascending order.

  3.3 − Next, it will choose the top K rows from the sorted array.

  3.4 − Now, it will assign a class to the test point based on most frequent class of these rows.

- Step 4 − End



```
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import accuracy_score
from sklearn.metrics import make_scorer
from sklearn.metrics import f1_score
from time import time

#Create a KNeighborsClassifire
knn = KNeighborsClassifier(n_neighbors=15)

#Train the model using the training sets
knn1 = knn.fit(X_train ,y_train)

#Predict the response time for test dataset
start=time()
y_pred1=knn1.predict(X_test)
end=time()
ts_time=end-start

print(confusion_matrix(y_test, y_pred1))
print(classification_report(y_test, y_pred1))

print("make prediction in {:.4f} second".format(ts_time))
f1_score=f1_score(y_test,y_pred1,pos_label='s')
knn_acc=accuracy_score(y_test,y_pred1)*100
score_dict['KNeighborsClassifier'] = knn_acc
print("f1 score and acuuracy score {:.4f},{:4f} ".format(f1_score,knn_acc))
```

```
[[11886  1960]
 [ 2440  4714]]
              precision    recall  f1-score   support

           b       0.83      0.86      0.84     13846
           s       0.71      0.66      0.68      7154

    accuracy                           0.79     21000
   macro avg       0.77      0.76      0.76     21000
weighted avg       0.79      0.79      0.79     21000

make prediction in 78.3087 second
f1 score and acuuracy score 0.6818,79.047619
```

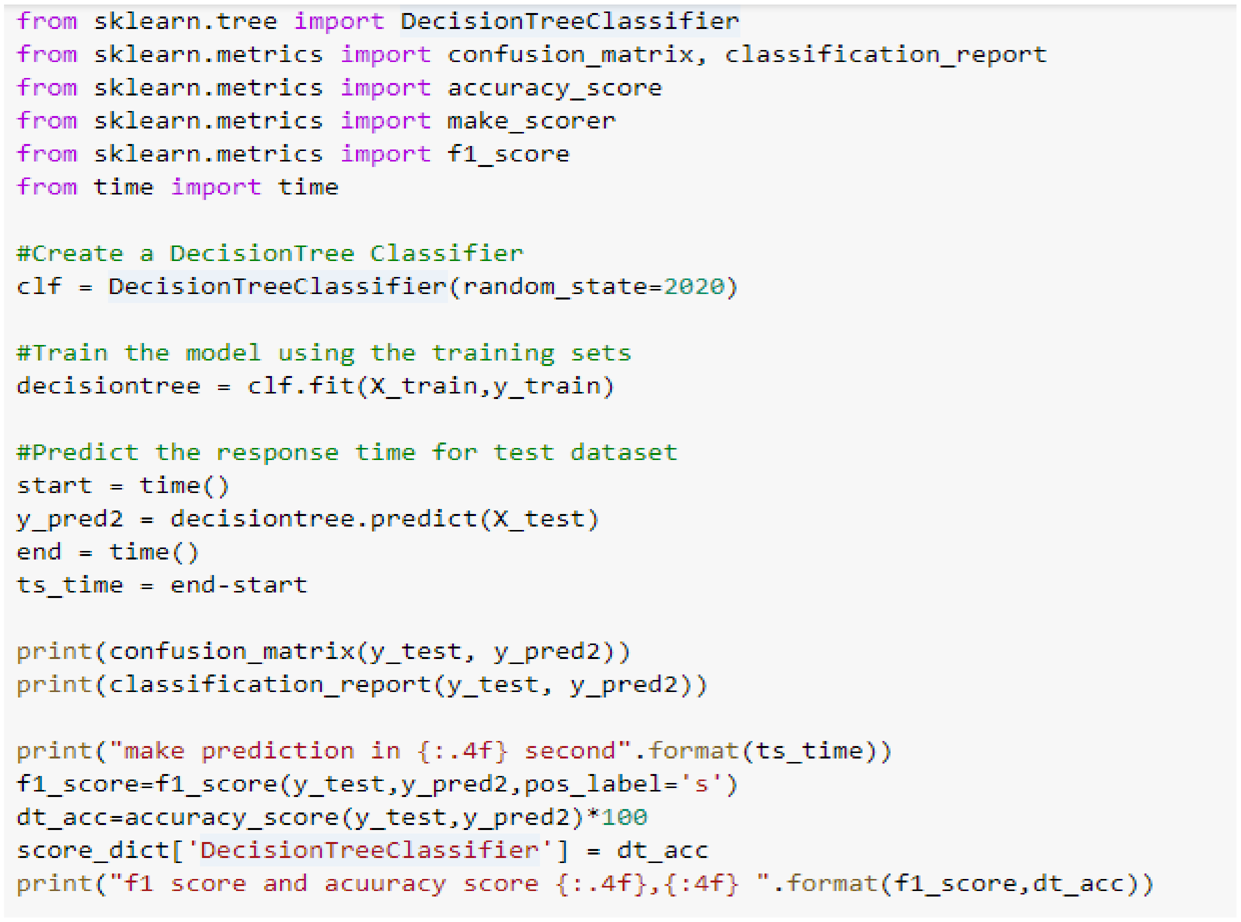
Fig. 6.2.1 F1 and accuracy score of KNeighborsClassifier

6.2.2 Decision Tree Model: Decision Trees create a set of binary splits on the predictor variables. These splits are used to classify new observations into one of the two groups. There are two categories of decision trees:

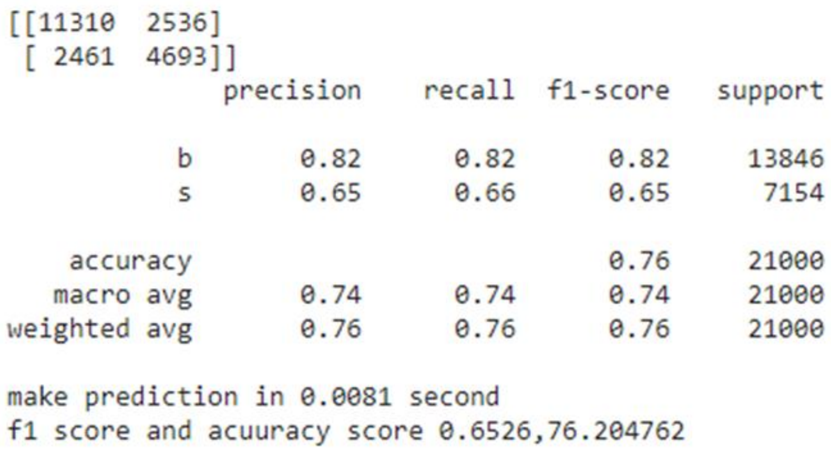
• Classification Tree for Categorical Response Variable

• Regression Tree for Numerical Response Variable

- **Step-1:** Begin the tree with the root node, says S, which contains the complete dataset.
- **Step-2:** Find the best attribute in the dataset using **Attribute Selection Measure (ASM).**
- **Step-3:** Divide the S into subsets that contains possible values for the best attributes.
- **Step-4:** Generate the decision tree node, which contains the best attribute.
- **Step-5:** Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.



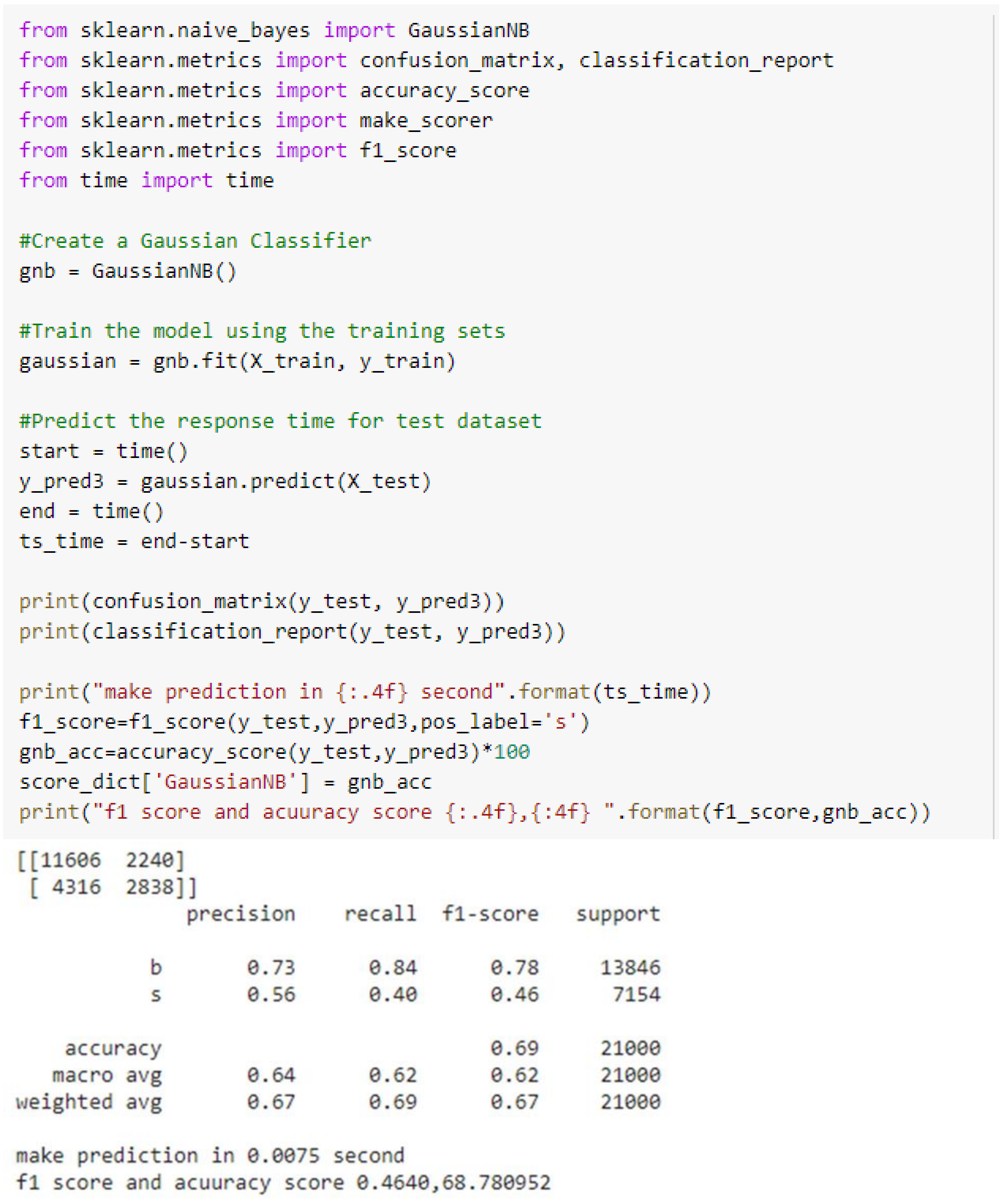
```python
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import accuracy_score
from sklearn.metrics import make_scorer
from sklearn.metrics import f1_score
from time import time

#Create a DecisionTree Classifier
clf = DecisionTreeClassifier(random_state=2020)

#Train the model using the training sets
decisiontree = clf.fit(X_train,y_train)

#Predict the response time for test dataset
start = time()
y_pred2 = decisiontree.predict(X_test)
end = time()
ts_time = end-start

print(confusion_matrix(y_test, y_pred2))
print(classification_report(y_test, y_pred2))

print("make prediction in {:.4f} second".format(ts_time))
f1_score=f1_score(y_test,y_pred2,pos_label='s')
dt_acc=accuracy_score(y_test,y_pred2)*100
score_dict['DecisionTreeClassifier'] = dt_acc
print("f1 score and acuuracy score {:.4f},{:4f} ".format(f1_score,dt_acc))
```



```
[[11310  2536]
 [ 2461  4693]]
              precision    recall  f1-score   support

           b       0.82      0.82      0.82     13846
           s       0.65      0.66      0.65      7154

    accuracy                           0.76     21000
   macro avg       0.74      0.74      0.74     21000
weighted avg       0.76      0.76      0.76     21000

make prediction in 0.0081 second
f1 score and acuuracy score 0.6526,76.204762
```

Fig. 6.2.2 F1 and accuracy score of DecisionTreeClassifier

6.2.3 Gaussian Naïve Bayes Model: It used for real-values (continuous) features. It assumes features follow a normal distribution.



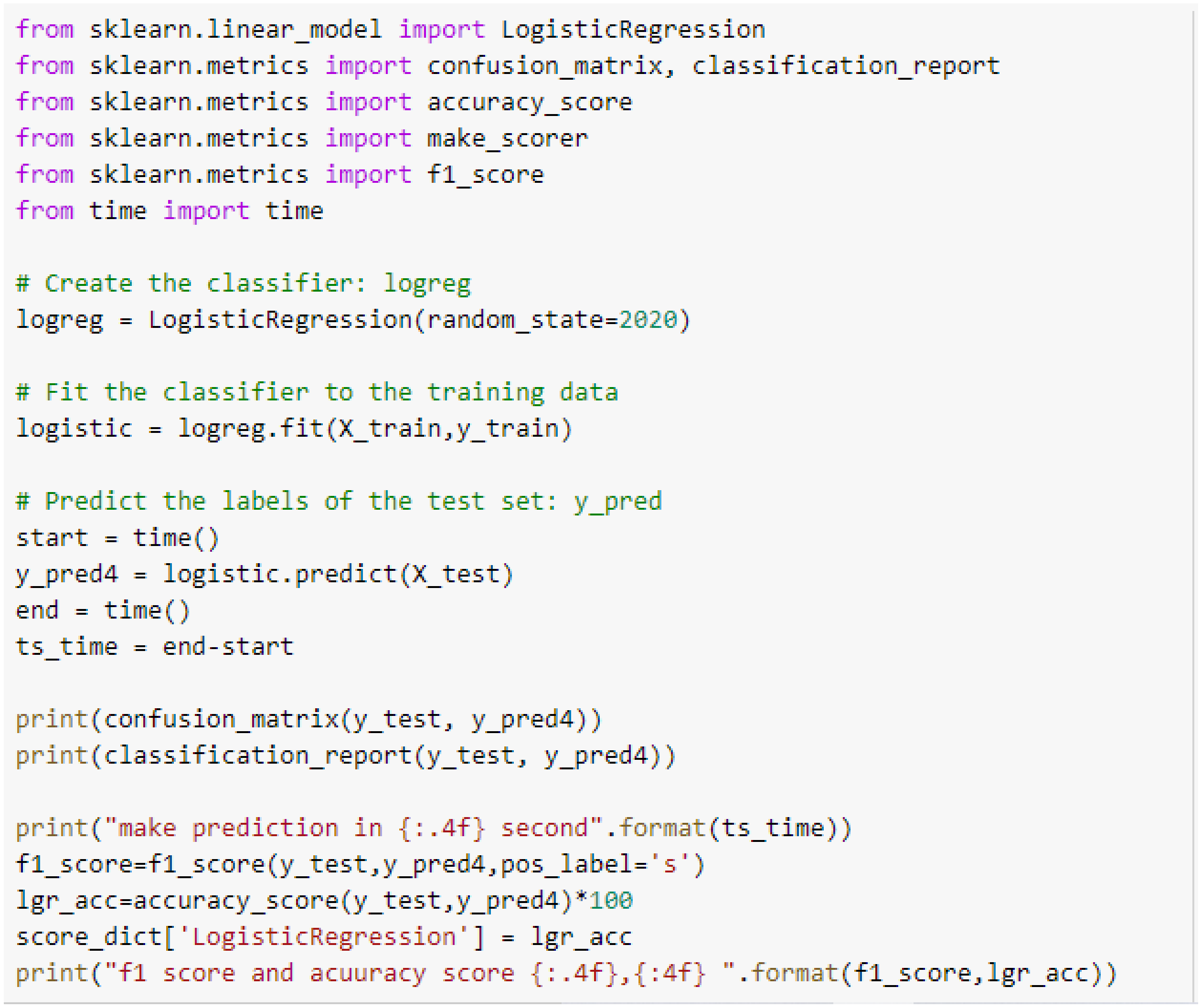
```python
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import accuracy_score
from sklearn.metrics import make_scorer
from sklearn.metrics import f1_score
from time import time

#Create a Gaussian Classifier
gnb = GaussianNB()

#Train the model using the training sets
gaussian = gnb.fit(X_train, y_train)

#Predict the response time for test dataset
start = time()
y_pred3 = gaussian.predict(X_test)
end = time()
ts_time = end-start

print(confusion_matrix(y_test, y_pred3))
print(classification_report(y_test, y_pred3))

print("make prediction in {:.4f} second".format(ts_time))
f1_score=f1_score(y_test,y_pred3,pos_label='s')
gnb_acc=accuracy_score(y_test,y_pred3)*100
score_dict['GaussianNB'] = gnb_acc
print("f1 score and acuuracy score {:.4f},{:4f} ".format(f1_score,gnb_acc))
```

```
[[11606  2240]
 [ 4316  2838]]
              precision    recall  f1-score   support

           b       0.73      0.84      0.78     13846
           s       0.56      0.40      0.46      7154

    accuracy                           0.69     21000
   macro avg       0.64      0.62      0.62     21000
weighted avg       0.67      0.69      0.67     21000

make prediction in 0.0075 second
f1 score and acuuracy score 0.4640,68.780952
```

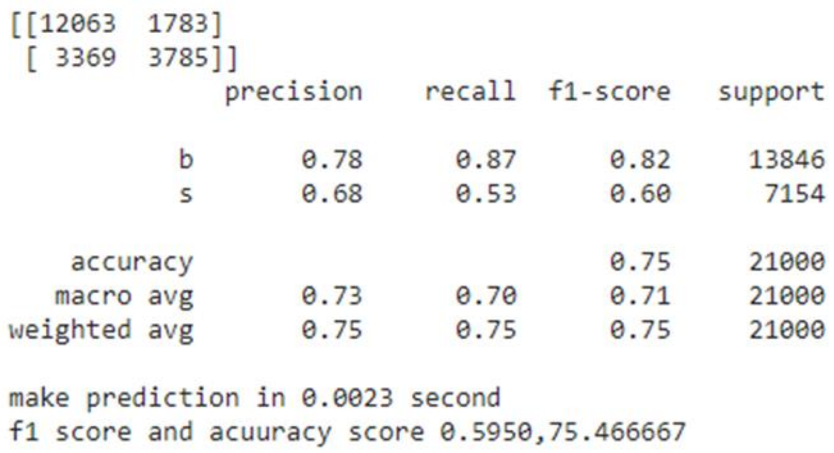
Fig. 6.2.3 F1 and accuracy score of GausianNB

6.2.4 Logistic Regression Model: It is used for predicting the categorical dependent variable using a given set of independent variables. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.

In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).



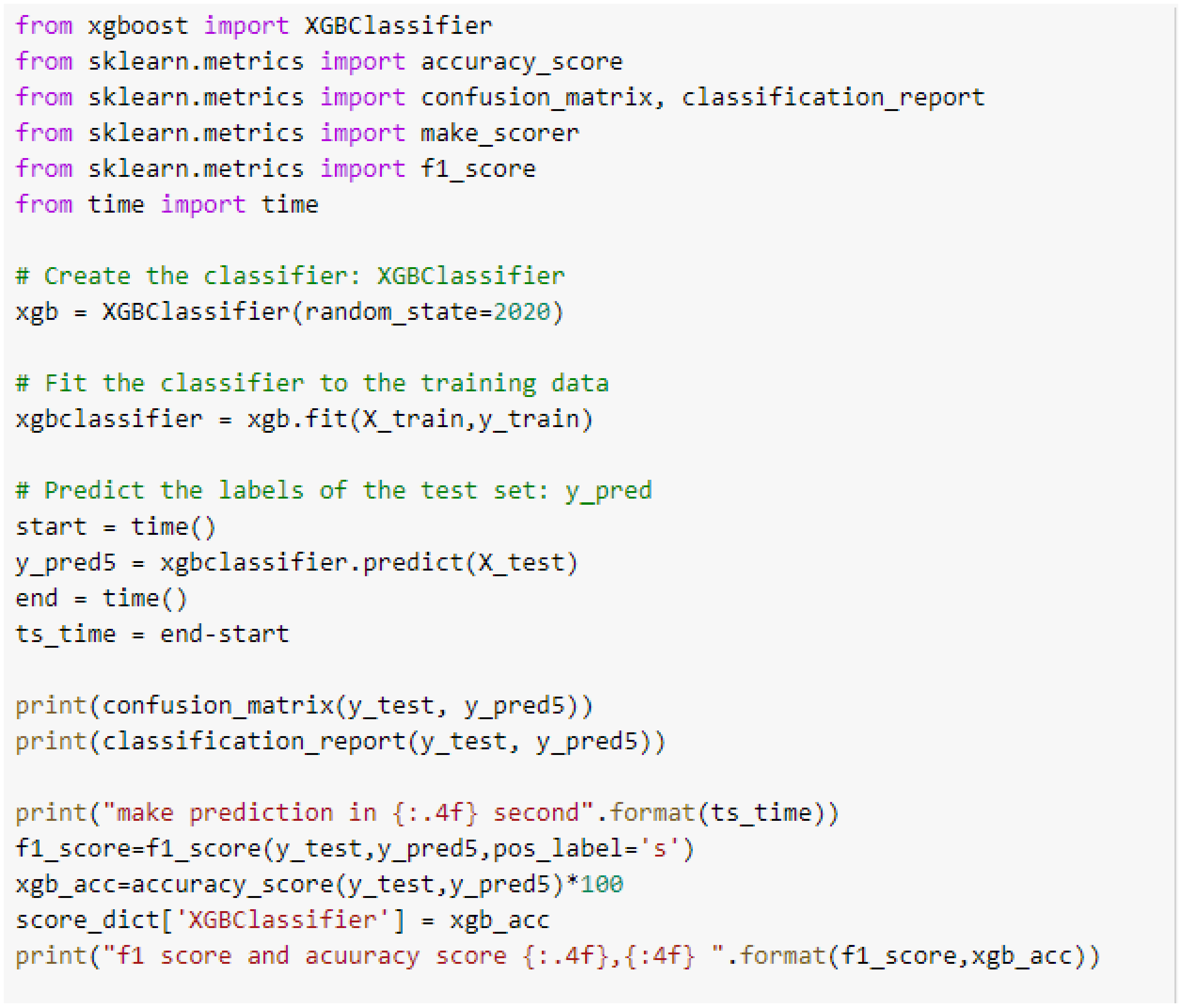
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import accuracy_score
from sklearn.metrics import make_scorer
from sklearn.metrics import f1_score
from time import time

# Create the classifier: logreg
logreg = LogisticRegression(random_state=2020)

# Fit the classifier to the training data
logistic = logreg.fit(X_train,y_train)

# Predict the labels of the test set: y_pred
start = time()
y_pred4 = logistic.predict(X_test)
end = time()
ts_time = end-start

print(confusion_matrix(y_test, y_pred4))
print(classification_report(y_test, y_pred4))

print("make prediction in {:.4f} second".format(ts_time))
f1_score=f1_score(y_test,y_pred4,pos_label='s')
lgr_acc=accuracy_score(y_test,y_pred4)*100
score_dict['LogisticRegression'] = lgr_acc
print("f1 score and acuuracy score {:.4f},{:4f} ".format(f1_score,lgr_acc))
```



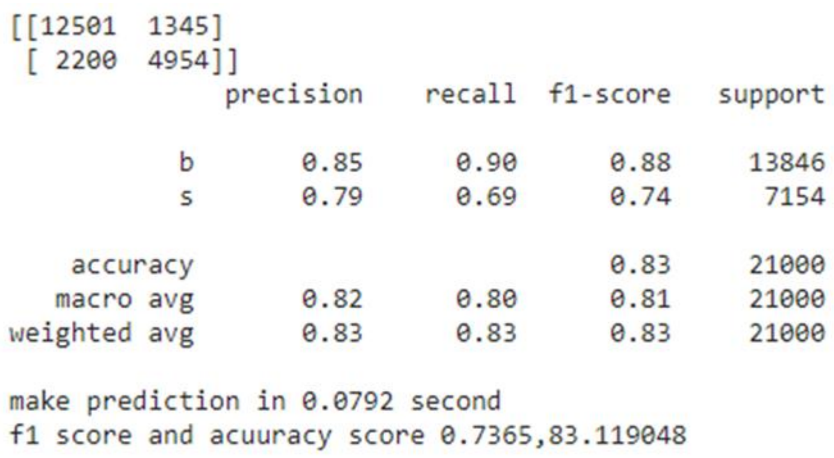
```
[[12063  1783]
 [ 3369  3785]]
              precision    recall  f1-score   support

           b       0.78      0.87      0.82     13846
           s       0.68      0.53      0.60      7154

    accuracy                           0.75     21000
   macro avg       0.73      0.70      0.71     21000
weighted avg       0.75      0.75      0.75     21000

make prediction in 0.0023 second
f1 score and acuuracy score 0.5950,75.466667
```

Fig. 6.2.4 F1 and accuracy score of Logistic Regression

6.2.5 Extreme Gradient Boosting Model (XGBClassifier): It is an additive and sequential model where trees are grown in sequential manner which converts weak learners into strong learners by adding weights to the weak learners and reduce weights of the strong learners. So each tree learns and boosts from the previous tree grown.



```python
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import make_scorer
from sklearn.metrics import f1_score
from time import time

# Create the classifier: XGBClassifier
xgb = XGBClassifier(random_state=2020)

# Fit the classifier to the training data
xgbclassifier = xgb.fit(X_train,y_train)

# Predict the labels of the test set: y_pred
start = time()
y_pred5 = xgbclassifier.predict(X_test)
end = time()
ts_time = end-start

print(confusion_matrix(y_test, y_pred5))
print(classification_report(y_test, y_pred5))

print("make prediction in {:.4f} second".format(ts_time))
f1_score=f1_score(y_test,y_pred5,pos_label='s')
xgb_acc=accuracy_score(y_test,y_pred5)*100
score_dict['XGBClassifier'] = xgb_acc
print("f1 score and acuuracy score {:.4f},{:4f} ".format(f1_score,xgb_acc))
```

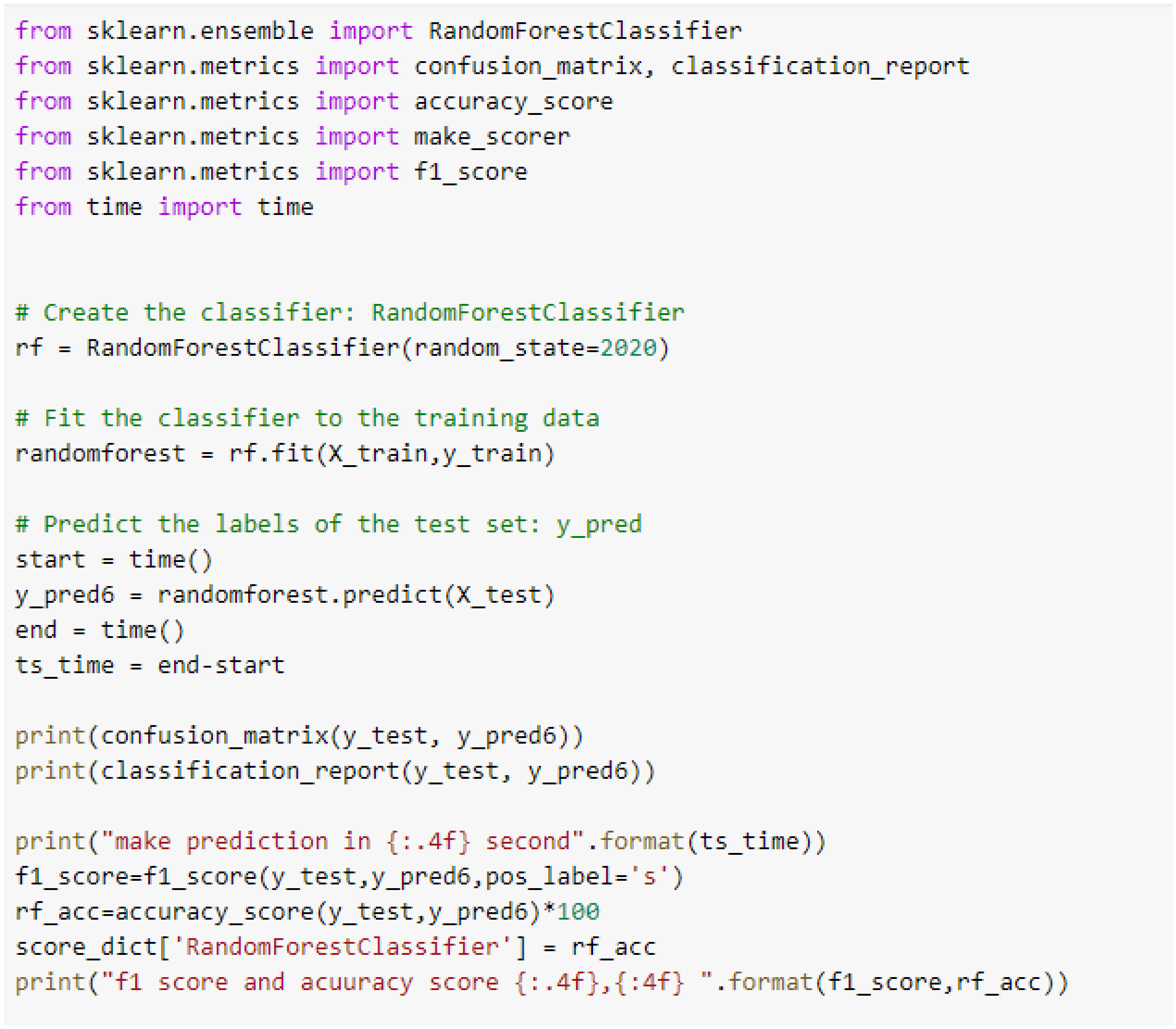


```
[[12501  1345]
 [ 2200  4954]]
              precision    recall  f1-score   support

           b       0.85      0.90      0.88     13846
           s       0.79      0.69      0.74      7154

    accuracy                           0.83     21000
   macro avg       0.82      0.80      0.81     21000
weighted avg       0.83      0.83      0.83     21000

make prediction in 0.0792 second
f1 score and acuuracy score 0.7365,83.119048
```

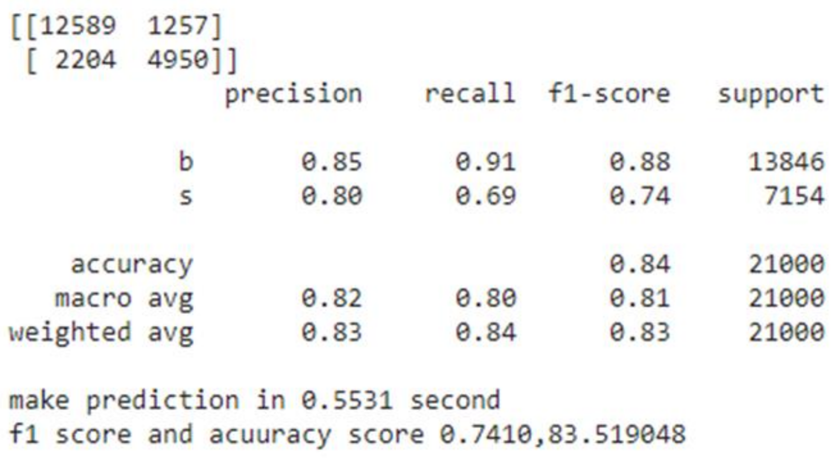
Fig. 6.2.5 F1 and accuracy score of Extreme Gradient Boosting (XGBClassifier)

6.2.6 Random Forest Model Model : In this algorithm, the observations as well as variables are sampled to create multiple decision trees Each observation is classified by each decision tree. The outcome is considered as per the majority
in different trees

- Step 1 − First, start with the selection of random samples from a given dataset.
- Step 2 − Next, this algorithm will construct a decision tree for every sample. Then it will get the prediction result from every decision tree.
- Step 3 − In this step, voting will be performed for every predicted result.
- Step 4 − At last, select the most voted prediction result as the final prediction result



```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import accuracy_score
from sklearn.metrics import make_scorer
from sklearn.metrics import f1_score
from time import time


# Create the classifier: RandomForestClassifier
rf = RandomForestClassifier(random_state=2020)

# Fit the classifier to the training data
randomforest = rf.fit(X_train,y_train)

# Predict the labels of the test set: y_pred
start = time()
y_pred6 = randomforest.predict(X_test)
end = time()
ts_time = end-start

print(confusion_matrix(y_test, y_pred6))
print(classification_report(y_test, y_pred6))

print("make prediction in {:.4f} second".format(ts_time))
f1_score=f1_score(y_test,y_pred6,pos_label='s')
rf_acc=accuracy_score(y_test,y_pred6)*100
score_dict['RandomForestClassifier'] = rf_acc
print("f1 score and acuuracy score {:.4f},{:4f} ".format(f1_score,rf_acc))
```



```
[[12589  1257]
 [ 2204  4950]]
              precision    recall  f1-score   support

           b       0.85      0.91      0.88     13846
           s       0.80      0.69      0.74      7154

    accuracy                           0.84     21000
   macro avg       0.82      0.80      0.81     21000
weighted avg       0.83      0.84      0.83     21000

make prediction in 0.5531 second
f1 score and acuuracy score 0.7410,83.519048
```

Fig. 6.2.5 F1 and accuracy score of Random Forest

6.3 Results:

The F1 Score and Accuracy Score were gathered by based on Six machine learning algorithms; k-NN, Decision Tree ,GaussianNB, Logistic Regression , XGBoost and Random Forest. The result of the data by using these techniques were documented in the following section. Table shows the final results for k-NN, Decision Tree ,GaussianNB, Logistic Regression , XGBoost and Random Forest.

Model	Precision	Recall	F1 Score	Accuracy Score
K-Nearest Neighbour	0.71	0.66	0.68	79.04
Decision tree	0.65	0.66	0.65	76.20
Gaussian Naïve Bayes	0.56	0.40	0.46	68.78
Logistic Regression	0.68	0.53	0.59	75.46
XGBoost	0.79	0.69	0.73	83.11
Random Forest	0.80	0.69	0.74	83.51

6.3.1 Table 1. Comparison of models

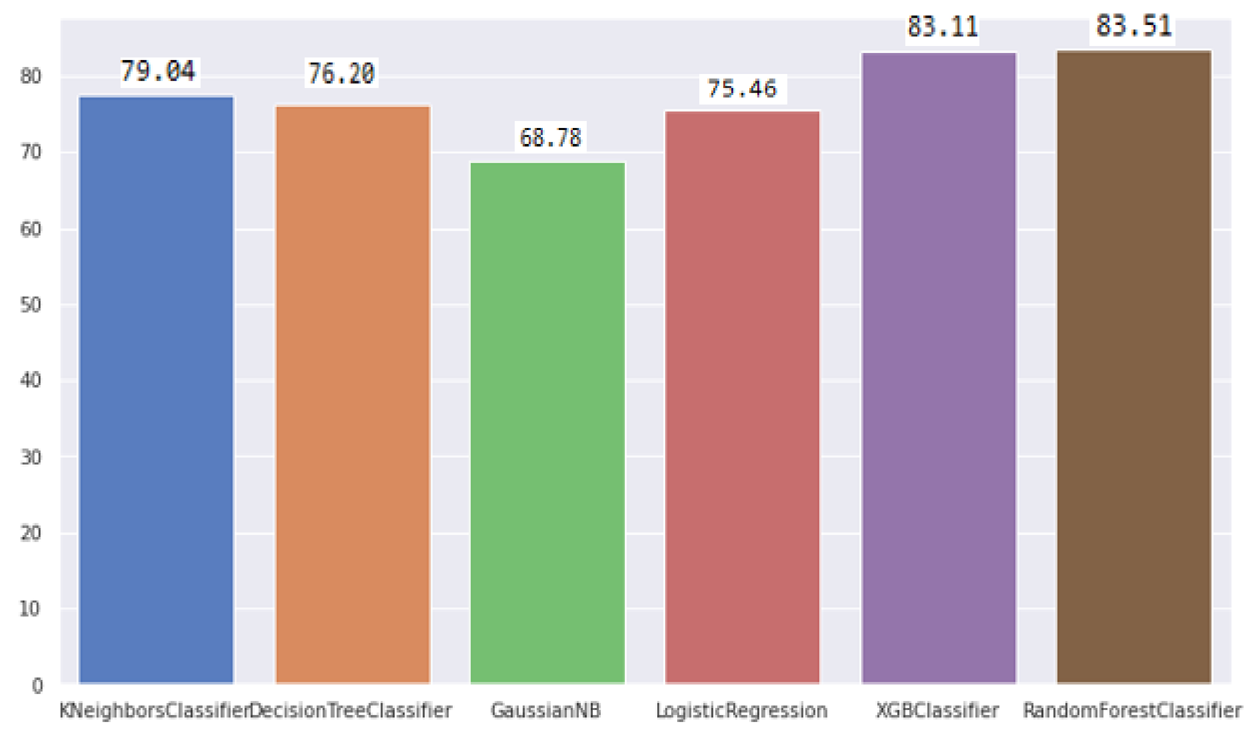


Fig. 6.3.2 Bar plots of Comparison models

CHAPTER – 7

CONCLUSION

Our proposed solution to the Higgs Boson Identification problem can be successfully implemented using the ML Approach. Further, we intend to finalize the results based on the Algorithm with maximum efficiency, i.e. selection on the basis of high Accuracy and F1 score. It would result in attaining the maximum Tau-Tau particles (signals), that eventually verifies the existence of Higgs-Boson Particle.

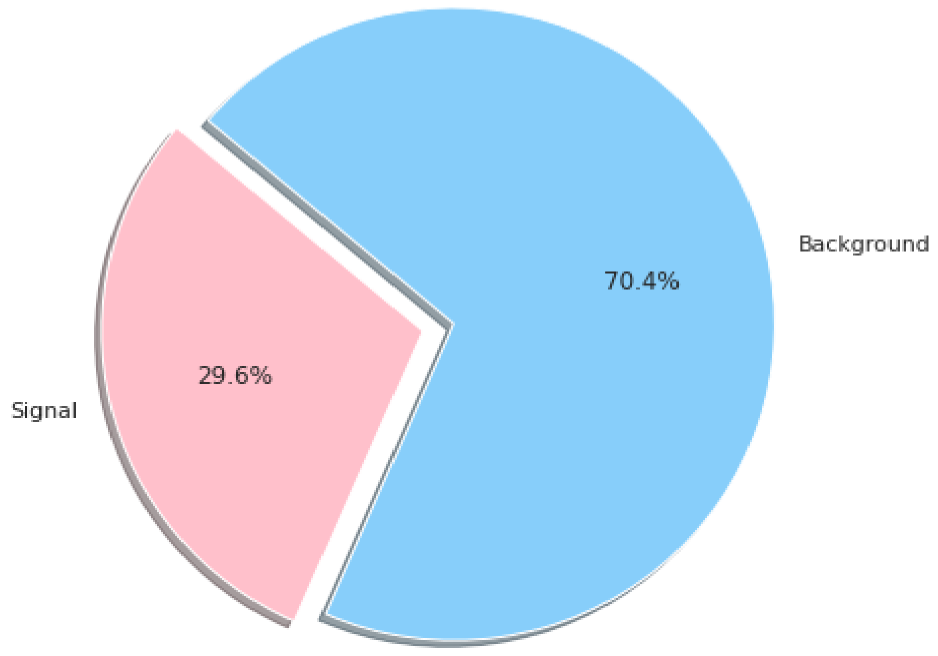


Fig. 7.1 Classification of Signal and Background Noise

.

CHAPTER – 8

Future Scope

Manufacturing :

By reducing the effect of the Higgs field on a metal with low mass metal working will be easier and the alloys formed can be more homogenous and of better quality.
By increasing the effect of the Higgs field, the mass of the metal can be increased allowing for high mass compaction of materials. Thus producing highly dense, stronger and more durable materials.

Transportation :

Transport of heavy construction equipment to remote locations would be both easier and speedier if the mass of the payload could be reduced.

Defence :

Explosive payloads could be transported to required sites at higher speeds and using lesser energy, if their mass could be reduced.
Low mass projectiles could be accelerated to higher speeds thus increasing their penetrative power seeking to cut through armor plates instead of striking against them

CHAPTER – 9

REFRENCES

[1] ATLAS collaboration (2014). “Dataset from the ATLAS Higgs Boson Machine Learning Challenge 2014.” CERN Open Data Portal. DOI:10.7483/OPENDATA.ATLAS.ZBP2.M5T8.

[2] Tom Cornelis. “Quark-gluon Jet Discrimination at CMS.” In Proceedings, 2nd Conference on Large Hadron Collider Physics Conference (LHCP 2014): New York, USA, June 2-7, 2014.

[3] C. Adam-Bourdarios, G. Cowan, C. Germain, I. Guyon, B. Kegl, and D. Rousseau. “Learning to discover: the higgs machine learning challenge 2014 - documentation.” CERN Open Data Portal, 2014. DOI:10.7483/OPENDATA.ATLAS.MQ5J.GHXA.

[4] S. Chatrchyan et al. “Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC.” Phys.Lett. DOI: 10.1016/j.physletb.2012.08.021.

[5] The ATLAS collaboration. “Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC.” Phys.Lett. B716 (2012) 1-29. arXiv:1207.7214. CERN-PH-EP-2012-218. DOI: 10.1016/j.physletb.2012.08.020.

[6] Safdari Hartman. “Neural Networks for calibrating ATLAS jets.” Phys.Lett. B659, pp. arXiv 1510.03823v1.pdf (2016).

[7] Matteo Cacciari, Gavin P. Salam. “Pileup subtraction using 514 jet areas.” Phys. Lett. B659, pp. 119126, (2008).

[8] Cukierman, Aviv and Benjamin Nachman. “Mathematical Properties of Numerical Inversion for Jet Calibrations.” arXiv:16.09.05195v1 [physics.data-an] 16 Sep 2016.