PROJECT REPORT

MACHINE LEARNING
(ARM-210)

BACHELOR OF TECHNOLOGY
in
Artificial Intelligence and Machine Learning
(4th Semester)

Submitted to:-
Dr. Amit Choudhary
Assistant Professor
USAR

Submitted by:-
Name: Aditya Maiti
Batch: AIML B1
Roll No:03819051622

University School of Automation and Robotics
GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY
(EAST DELHI CAMPUS)
SURAJMAL VIHAR, NEW DELHI-110032

BASIC INFORMATION

Title of the Project- Classification on Magic Gamma Telescope Dataset
Student Name- Aditya Maiti
Enrollment No.- 03819051622
Email- adityamaiti.123@gmail.com
Contact- 9318366526
Google Drive Link-
https://drive.google.com/drive/folders/1n6AL2h5aGDDp3EVZOlnx4sNit8gX8cUg?usp=drive_link

REPORT

TITLE-

Classification on Magic Gamma Telescope Dataset

ABSTRACT-

The ground-based MAGIC telescopes are essential tools for researching high-energy gamma rays. It is difficult to distinguish these gamma rays from hadronic background noise, though. The dataset utilised in this study is artificial data created by Monte Carlo simulations to imitate the registration process of charged particles released in electromagnetic showers that are started by gamma rays in the atmosphere.

In this project, the MAGIC Telescope dataset's gamma/hadron classification is studied using machine learning techniques. Six models of classification were used by us: (name the six models you used). Preprocessing was done on the dataset to fix any imbalances and get it ready for model training. Accuracy served as the main performance indicator for each model. The findings showed that in terms of separating gamma rays from hadrons, (name the model with the highest accuracy) had the best accuracy of (name the accuracy percentage).

This project contributes to a deeper understanding of high-energy astrophysical phenomena by showcasing the potential of machine learning to improve gamma-ray identification in the MAGIC Telescope data.

KEYWORDS-

1. Classification
2. KNN Classifier
3. SVM
4. Decision Trees
5. Linear Regression
6. Random Forests
7. Gamma Rays
8. Hadron Background
9. Cherenkov Radiations

INTRODUCTION-

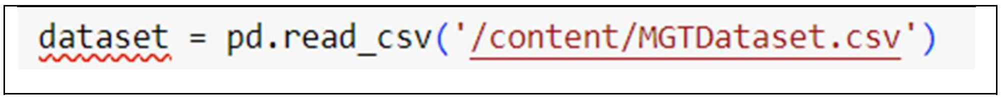
The process of classifying input data into predefined classes or labels based on their features is known as classification in machine learning. In this project we'll do a brief analysis of the dataset, and then use 6 classifiers on our real time dataset-

1. KNN Classifier
2. SVM
3. Decision Trees
4. Linear Regression
5. Random Forests
6. Naïve Bayes Classifier

We'll compare the accuracy of these models and develop predictive models that can accurately assign class labels to new instances.

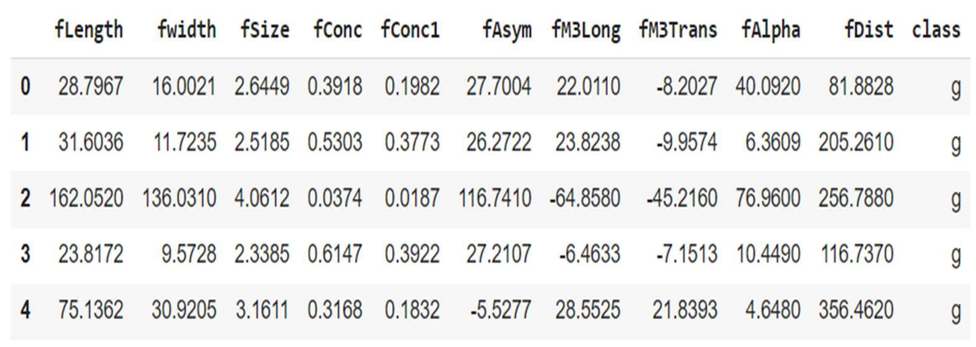
PROPOSED METHADOLOGY-

1. **Importing Dependencies-** This refers to incorporating external libraries or modules that provide functionalities for our project needs.
2. **Importing the dataset-** This involves incorporating our real time dataset into our project by making use of .csv file.



```
dataset = pd.read_csv('/content/MGTDataset.csv')
```

This line of code reads the data from the CSV file 'MGTDataset.csv' located at the specified path and stores it in a DataFrame variable named dataset.

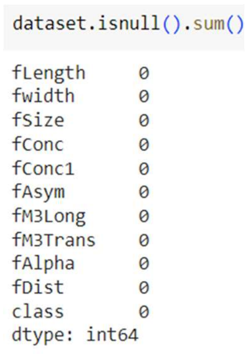


	fLength	fwidth	fSize	fConc	fConc1	fAsym	fM3Long	fM3Trans	fAlpha	fDist	class
0	28.7967	16.0021	2.6449	0.3918	0.1982	27.7004	22.0110	-8.2027	40.0920	81.8828	g
1	31.6036	11.7235	2.5185	0.5303	0.3773	26.2722	23.8238	-9.9574	6.3609	205.2610	g
2	162.0520	136.0310	4.0612	0.0374	0.0187	116.7410	-64.8580	-45.2160	76.9600	256.7880	g
3	23.8172	9.5728	2.3385	0.6147	0.3922	27.2107	-6.4633	-7.1513	10.4490	116.7370	g
4	75.1362	30.9205	3.1611	0.3168	0.1832	-5.5277	28.5525	21.8393	4.6480	356.4620	g

DATASET

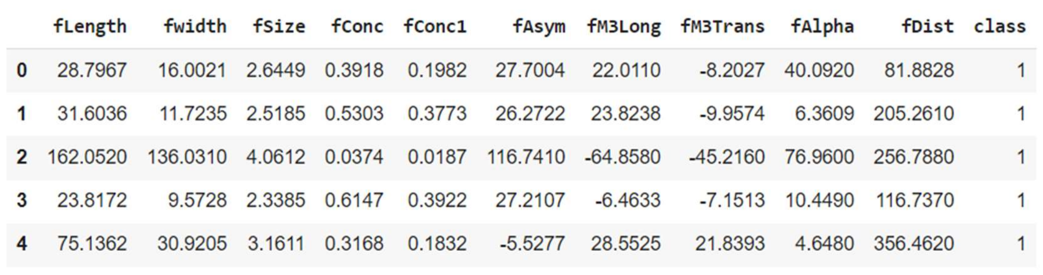
3. **Data preprocessing-**

This involves checking if there are any missing values. If missing values are present they’re usually handled by deleting the record. In this case no missing values are found.



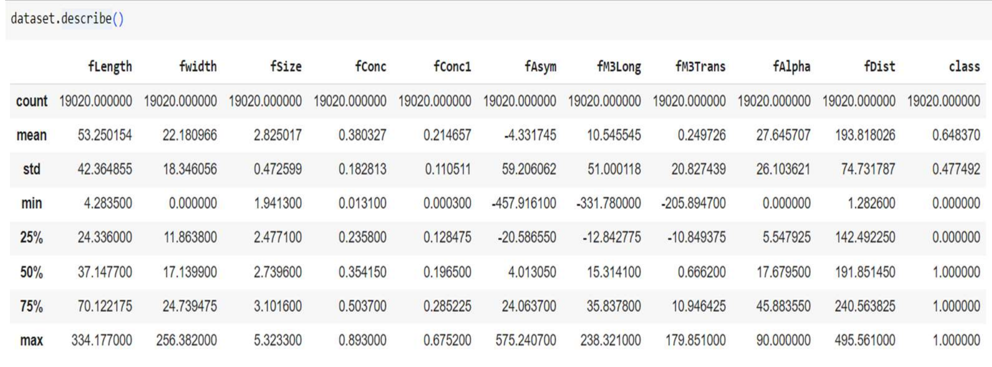
```
dataset.isnull().sum()

fLength     0
fwidth      0
fSize       0
fConc       0
fConc1      0
fAsym       0
fM3Long     0
fM3Trans    0
fAlpha      0
fDist       0
class       0
dtype: int64
```

Data Preprocessing also involves converting categorical data into numerical data. In this case changes will be made in the last column. The dataset now-



	fLength	fwidth	fSize	fConc	fConc1	fAsym	fM3Long	fM3Trans	fAlpha	fDist	class
0	28.7967	16.0021	2.6449	0.3918	0.1982	27.7004	22.0110	-8.2027	40.0920	81.8828	1
1	31.6036	11.7235	2.5185	0.5303	0.3773	26.2722	23.8238	-9.9574	6.3609	205.2610	1
2	162.0520	136.0310	4.0612	0.0374	0.0187	116.7410	-64.8580	-45.2160	76.9600	256.7880	1
3	23.8172	9.5728	2.3385	0.6147	0.3922	27.2107	-6.4633	-7.1513	10.4490	116.7370	1
4	75.1362	30.9205	3.1611	0.3168	0.1832	-5.5277	28.5525	21.8393	4.6480	356.4620	1

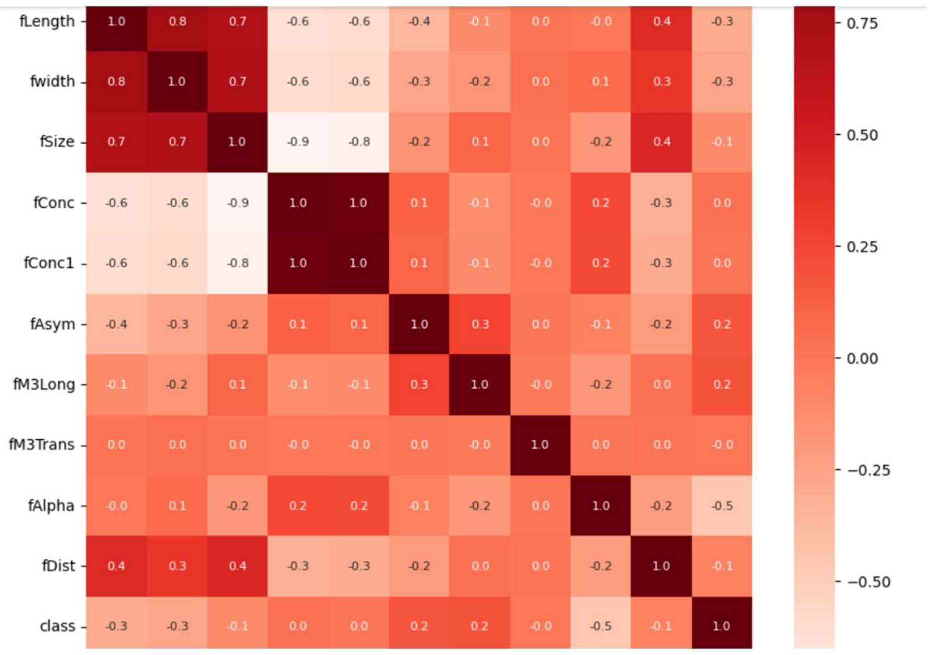
4. **Data description-** We make use of DataFrame objects in pandas so that it produces descriptive statistics that highlight the distribution's shape, central tendency, and dispersion.



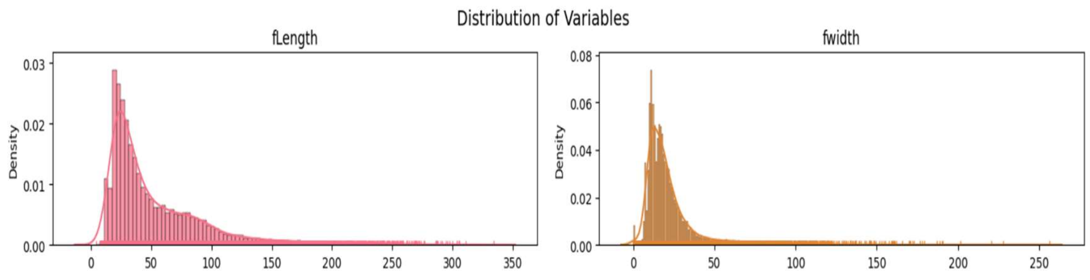
```
dataset.describe()
```

	fLength	fwidth	fSize	fConc	fConc1	fAsym	fM3Long	fM3Trans	fAlpha	fDist	class
count	19020.000000	19020.000000	19020.000000	19020.000000	19020.000000	19020.000000	19020.000000	19020.000000	19020.000000	19020.000000	19020.000000
mean	53.250154	22.180966	2.825017	0.380327	0.214657	-4.331745	10.545545	0.249726	27.645707	193.818026	0.648370
std	42.364855	18.346056	0.472599	0.182813	0.110511	59.206062	51.000118	20.827439	26.103621	74.731787	0.477492
min	4.283500	0.000000	1.941300	0.013100	0.000300	-457.916100	-331.780000	-205.894700	0.000000	1.282600	0.000000
25%	24.336000	11.863800	2.477100	0.235800	0.128475	-20.586550	-12.842775	-10.849375	5.547925	142.492250	0.000000
50%	37.147700	17.139900	2.739600	0.354150	0.196500	4.013050	15.314100	0.666200	17.679500	191.851450	1.000000
75%	70.122175	24.739475	3.101600	0.503700	0.285225	24.063700	35.837800	10.946425	45.883550	240.563825	1.000000
max	334.177000	256.382000	5.323300	0.893000	0.675200	575.240700	238.321000	179.851000	90.000000	495.561000	1.000000

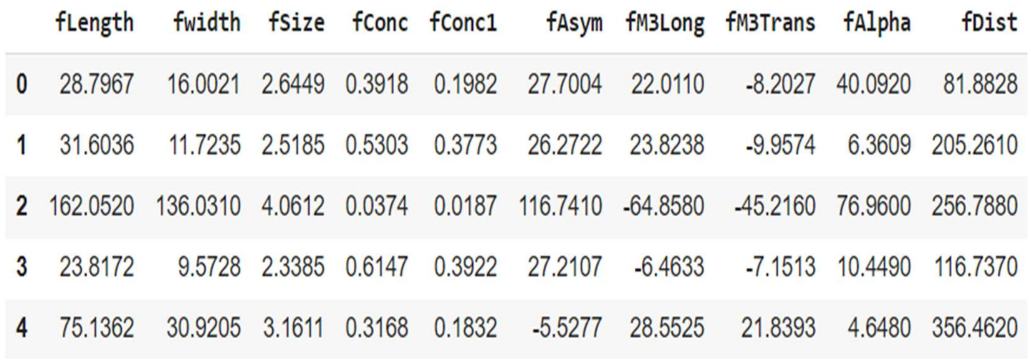
5. **Plot-** We make use of matplotlib.pyplot and seaborn to make a barplot of 'class vs fwidth' and 'class vs fconc1'. As shown below we also make a heatmap to look at the necessary and unnecessary data.



6. **Distribution of values-** This code segment generates a grid of subplots, each of which uses rugplots, histograms, and kernel density estimation (KDE) to visualise the distribution of values for a particular attribute in the dataset.

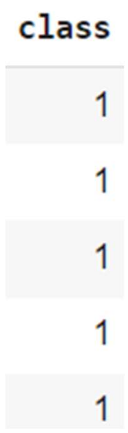


7. **Splitting the Dataset-** The dataset will be split into X and Y, Y will contain the last column and X will contain the rest of the columns. Then X is further split into training data which is used for training the model and testing data which is used to test the model.



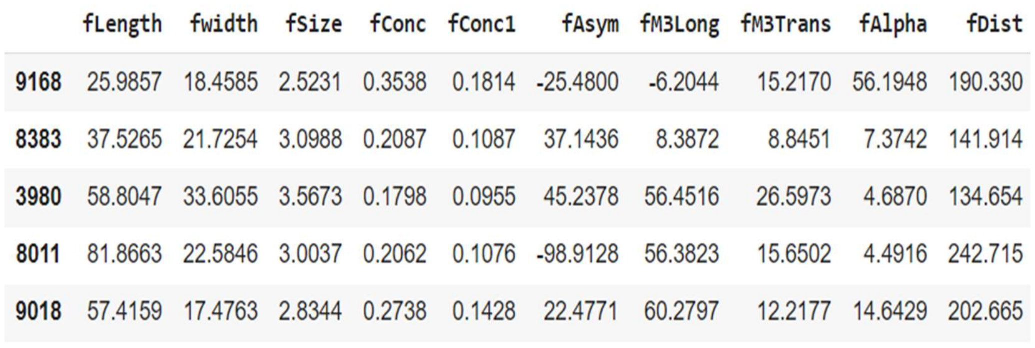
	fLength	fwidth	fSize	fConc	fConc1	fAsym	fM3Long	fM3Trans	fAlpha	fDist
0	28.7967	16.0021	2.6449	0.3918	0.1982	27.7004	22.0110	-8.2027	40.0920	81.8828
1	31.6036	11.7235	2.5185	0.5303	0.3773	26.2722	23.8238	-9.9574	6.3609	205.2610
2	162.0520	136.0310	4.0612	0.0374	0.0187	116.7410	-64.8580	-45.2160	76.9600	256.7880
3	23.8172	9.5728	2.3385	0.6147	0.3922	27.2107	-6.4633	-7.1513	10.4490	116.7370
4	75.1362	30.9205	3.1611	0.3168	0.1832	-5.5277	28.5525	21.8393	4.6480	356.4620

X



class
1
1
1
1
1

Y



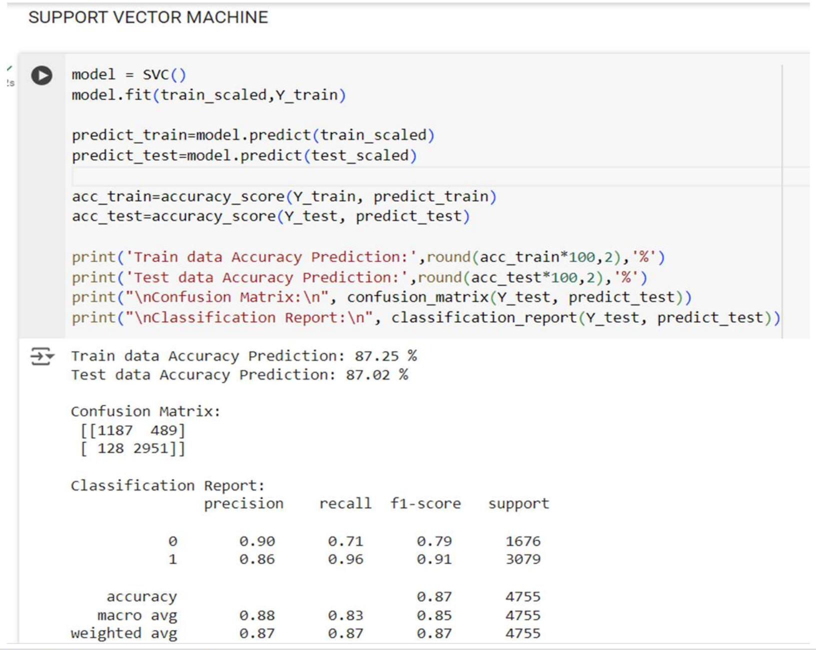
	fLength	fwidth	fSize	fConc	fConc1	fAsym	fM3Long	fM3Trans	fAlpha	fDist
9168	25.9857	18.4585	2.5231	0.3538	0.1814	-25.4800	-6.2044	15.2170	56.1948	190.330
8383	37.5265	21.7254	3.0988	0.2087	0.1087	37.1436	8.3872	8.8451	7.3742	141.914
3980	58.8047	33.6055	3.5673	0.1798	0.0955	45.2378	56.4516	26.5973	4.6870	134.654
8011	81.8663	22.5846	3.0037	0.2062	0.1076	-98.9128	56.3823	15.6502	4.4916	242.715
9018	57.4159	17.4763	2.8344	0.2738	0.1428	22.4771	60.2797	12.2177	14.6429	202.665

X_train

8. **Scaling(Standardisation)-** To ensure that the data is prepared for training and evaluation, this code segment preprocesses the training and testing feature data by standardising their scales using the scikit-learn StandardScaler.

```
X_train_array = np.asarray(X_train)
X_test_array = np.asarray(X_test)
sc=StandardScaler() # Machine Instance
train_scaled=sc.fit_transform(X_train_array) # Scaling the train set
test_scaled=sc.transform(X_test_array) # Scaling the test set
train_scaled
```

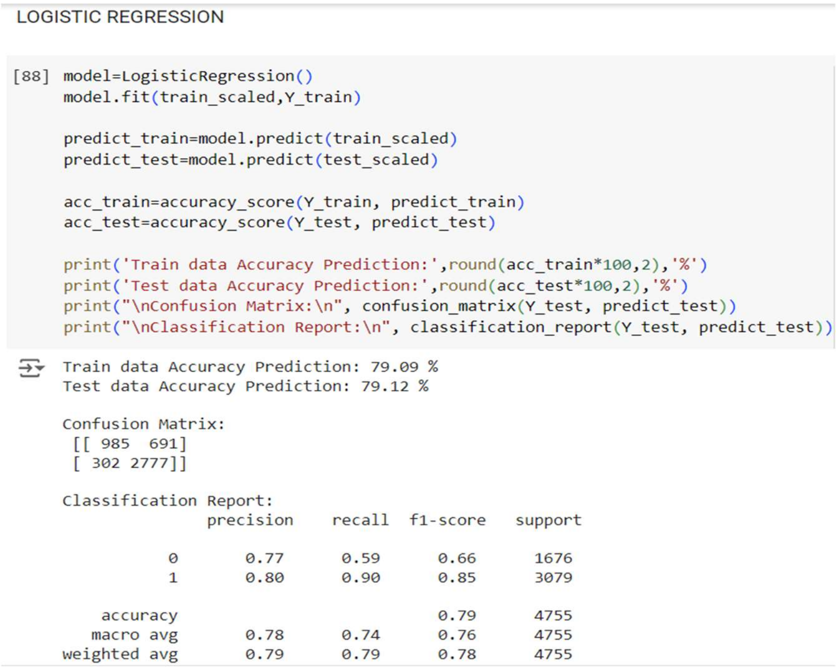
9. **Accuracy prediction via 6 models-** The following sections of code trains the corresponding classification model, assesses how well it performs on training and testing datasets, and offers a thorough breakdown of its classification accuracy. They also provide the confusion matrix and the classification report. Here example of SVM and logistic regression is given.



SUPPORT VECTOR MACHINE

```
model = SVC()
model.fit(train_scaled,Y_train)

predict_train=model.predict(train_scaled)
predict_test=model.predict(test_scaled)

acc_train=accuracy_score(Y_train, predict_train)
acc_test=accuracy_score(Y_test, predict_test)

print('Train data Accuracy Prediction:',round(acc_train*100,2),'%')
print('Test data Accuracy Prediction:',round(acc_test*100,2),'%')
print("\nConfusion Matrix:\n", confusion_matrix(Y_test, predict_test))
print("\nClassification Report:\n", classification_report(Y_test, predict_test))
```

```
Train data Accuracy Prediction: 87.25 %
Test data Accuracy Prediction: 87.02 %

Confusion Matrix:
 [[1187  489]
 [ 128 2951]]

Classification Report:
               precision    recall  f1-score   support

           0       0.90      0.71      0.79      1676
           1       0.86      0.96      0.91      3079

    accuracy                           0.87      4755
   macro avg       0.88      0.83      0.85      4755
weighted avg       0.87      0.87      0.87      4755
```

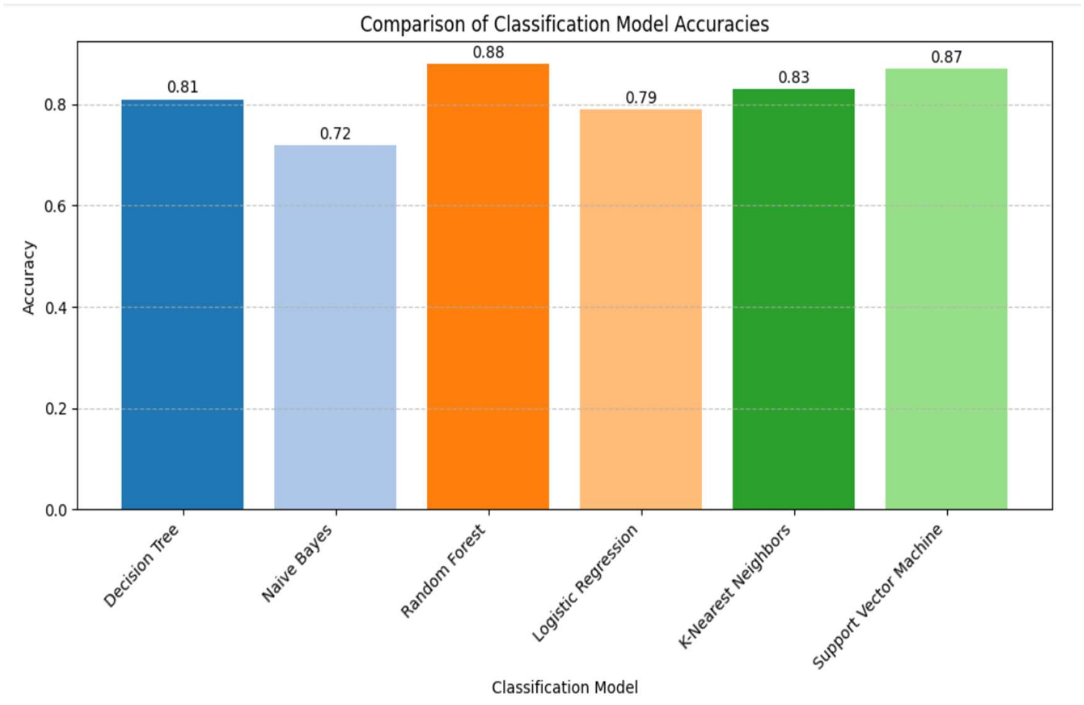


LOGISTIC REGRESSION

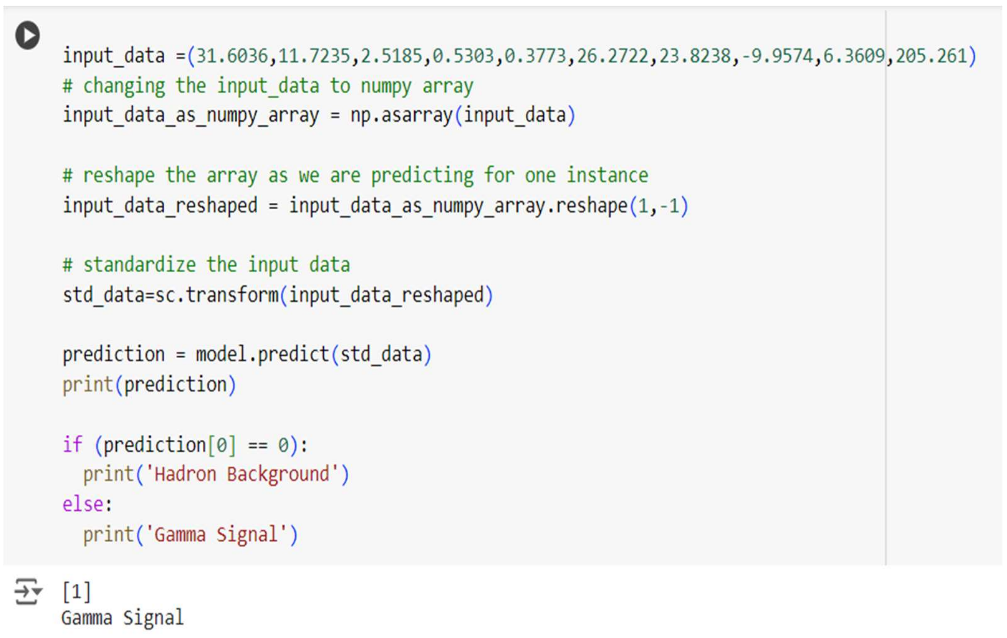
```
[88] model=LogisticRegression()
     model.fit(train_scaled,Y_train)

     predict_train=model.predict(train_scaled)
     predict_test=model.predict(test_scaled)

     acc_train=accuracy_score(Y_train, predict_train)
     acc_test=accuracy_score(Y_test, predict_test)

     print('Train data Accuracy Prediction:',round(acc_train*100,2),'%')
     print('Test data Accuracy Prediction:',round(acc_test*100,2),'%')
     print("\nConfusion Matrix:\n", confusion_matrix(Y_test, predict_test))
     print("\nClassification Report:\n", classification_report(Y_test, predict_test))
```

```
Train data Accuracy Prediction: 79.09 %
Test data Accuracy Prediction: 79.12 %

Confusion Matrix:
 [[ 985  691]
 [ 302 2777]]

Classification Report:
               precision    recall  f1-score   support

           0       0.77      0.59      0.66      1676
           1       0.80      0.90      0.85      3079

    accuracy                           0.79      4755
   macro avg       0.78      0.74      0.76      4755
weighted avg       0.79      0.79      0.78      4755
```

10. **Comparison of the 6 models –** In this section of we quickly compare the performance of various classification models by creating an eye-catching bar chart that compares their accuracies. We make use of numpy and matplotlib.



11. **Making a Predictive Model-** This section of code takes a data point, preprocesses it by standardising it with the same scaler that was used for training, and then uses a machine learning model that has been trained to predict the class label. Lastly, it prints the anticipated class label and a message corresponding to the classification outcome.



```
input_data =(31.6036,11.7235,2.5185,0.5303,0.3773,26.2722,23.8238,-9.9574,6.3609,205.261)
# changing the input_data to numpy array
input_data_as_numpy_array = np.asarray(input_data)

# reshape the array as we are predicting for one instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

# standardize the input data
std_data=sc.transform(input_data_reshaped)

prediction = model.predict(std_data)
print(prediction)

if (prediction[0] == 0):
  print('Hadron Background')
else:
  print('Gamma Signal')
```

```
[1]
Gamma Signal
```

RESULT AND DISCUSSION-

CLASSIFICATION MODELS	TESTING DATA ACCURACY
SVM	0.87
KNN	0.83
Logistic Regression	0.79
Random Forest	0.88
Naïve Bayes	0.72
Decision Tree	0.81

Ultimately it can deciphered from the table above and the comparison bar chart that Naïve Bayes classification model is the least accurate(0.72) while Random Forest Classification Model is the most accurate(0.88).

CONCLUSION AND FUTURE WORK-

Through this project I was able to obtain practical experience with a range of machine learning concepts. Important lessons learned include:

Learning Experience: Using simulated data, the project gave participants hands-on experience creating and evaluating classification models.

Preprocessing Techniques: I discovered how crucial feature scaling and data normalisation are to efficient model training.

Model Selection: I became more knowledgeable about model selection through experimenting with various classification algorithms, such as Decision Trees, Naive Bayes, Random Forest, Logistic Regression, K-Nearest Neighbours, and Support Vector Machines.

Evaluation Metrics: In order to evaluate the performance of the model, it was essential to comprehend evaluation metrics like accuracy, precision, recall, F1-score, confusion matrix, and classification report.

Data Visualisation: Model performance comparisons and insights into variable distributions were made possible through the use of data visualisation techniques.

My goal is to investigate more sophisticated machine learning techniques in the future, such as feature selection, ensemble methods, hyperparameter tuning, and interpretability of the model. This project establishes the groundwork for developing artificial intelligence and solving real-world problems in a variety of domains.

REFERENCES-

1. Python Machine Learning By Dr. Randal S. Olson
2. https://www.youtube.com/watch?v=gmvvaobm7eQ&list=PLeo1K3hjS3uvCeTYTeyfe0-rN5r8zn9rw
3. https://www.youtube.com/watch?v=bY__YW-xknU&list=PLfFghEzKVmjsNtIRwErklMAN8nJmebB0I
4. https://ieeexplore.ieee.org/abstract/document/8950650
5. https://www.sciencedirect.com/science/article/abs/pii/S016740480200514X
6. https://see.xidian.edu.cn/faculty/chzheng/bishe/indexfiles/new_folder/svm.pdf
7. https://www.sciencedirect.com/science/article/abs/pii/S0957417419303574
8. https://faculty.ksu.edu.sa/sites/default/files/classification_of_large_datasets_using_random.pdf
9. https://www.sciencedirect.com/science/article/abs/pii/S0950705117301880
10. https://gemini.google.com/app
11. https://books.google.co.in/books?hl=en&lr=&id=IvAw_1MTASsC&oi=fnd&pg=PA3&dq=Linear+Regression+&ots=GueHaq8wqD&sig=lv4mfdb8v2vU0mJlVGjP2TeD4LQ&redir_esc=y#v=onepage&q=Linear%20Regression&f=false
12. https://drive.google.com/file/d/1W-2YrtwnGqwJpVzH3z57zseilow_ax6k/view
13. https://drive.google.com/file/d/1G_R4T_Osvn-uB0IwTq_BNjxlhhYf1Dp-/view