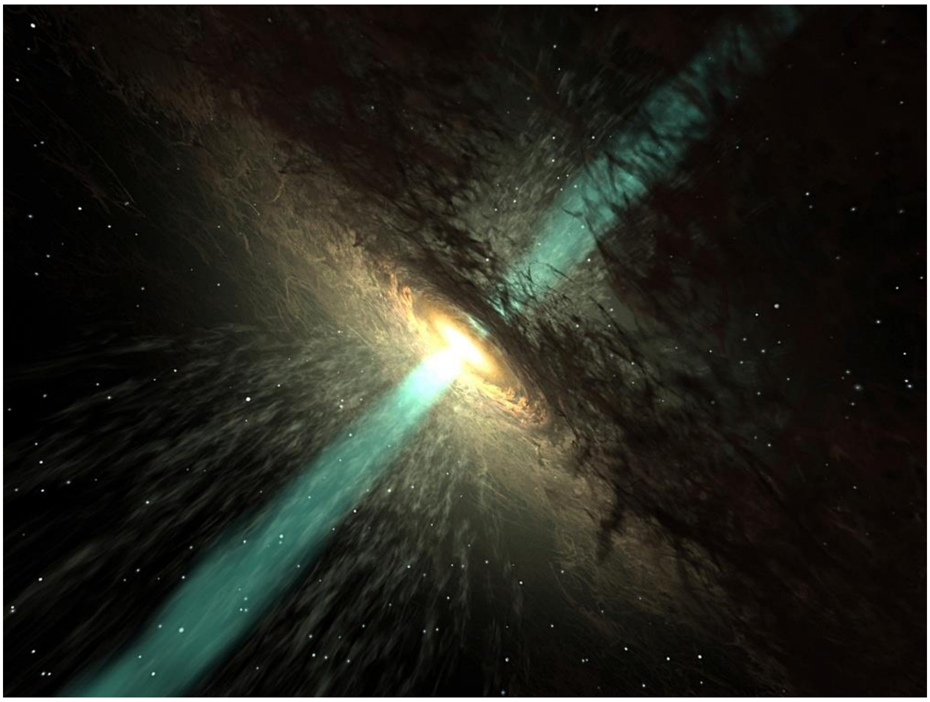
DETECTING PULSAR STARS

IDC-301 ENDSEM PROJECT



RUPAM KUNDU

Dept. of Physical Science

IISER Berhampur

DETECTING PULSAR STAR USING MACHINE LEARNING ALGORITHM

In this project I will explain whether a star is a pulsar or not, using mainly Logistic Regression. This project is readily available online.

INTRODUCTION

Pulsars are a rare type of Neutron star that produce radio emission detectable here on Earth. They are of considerable scientific interest as probes of space-time, the inter-stellar medium, and states of matter. Neutron stars are very dense, and have short, regular rotational periods. This produces a very precise interval between pulses that ranges from milliseconds to seconds for an individual pulsar. Pulsars are believed to be one of the candidates for the source of ultra-high-energy cosmic rays.

The first pulsar was observed on November 28, 1967, by Jocelyn Bell Burnell and Antony Hewish. They observed pulses separated by 1.33 seconds that originated from the same location in the sky, and kept to sidereal time. In looking for explanations for the pulses, the short period of the pulses eliminated most astrophysical sources of radiation, such as stars, and since the pulses followed sidereal time, it could not be man-made radio frequency interference.

PROJECT PLAN

1. Importing libraries needed
2. Collecting the Data
3. Analyzing Data

[-Producing a heat-map for correlation between different features
-Producing pair-plot to show correlation between features with classes]

4. Data wrangling and Testing Data
5. Finding Accuracy and Building a Confusion Matrix

1. Importing libraries needed

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

We have included some useful libraries. Such as; NumPy, pandas, matplotlib, seaborn.

2. Collecting the Data

I have collected the data from GitHub. For checking the website, you can click here.

In [3]:
```
data=pd.read_csv(r"C:\Users\Rupam\Desktop\pulsar_stars.csv")
```

In [4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17898 entries, 0 to 17897
Data columns (total 9 columns):
 #   Column                                         Non-Null Count  Dtype  
---  ------                                         --------------  -----  
 0    Mean of the integrated profile               17898 non-null  float64
 1    Standard deviation of the integrated profile 17898 non-null  float64
 2    Excess kurtosis of the integrated profile    17898 non-null  float64
 3    Skewness of the integrated profile           17898 non-null  float64
 4    Mean of the DM-SNR curve                     17898 non-null  float64
 5    Standard deviation of the DM-SNR curve       17898 non-null  float64
 6    Excess kurtosis of the DM-SNR curve          17898 non-null  float64
 7    Skewness of the DM-SNR curve                 17898 non-null  float64
 8   target_class                                  17898 non-null  int64  
dtypes: float64(8), int64(1)
memory usage: 1.2 MB
```

It contains 9 columns and 17898 rows initially.

The rows are:

1. mean_integrated_profile
2. std_deviation_integrated_profile
3. kurtosis_integrated_profile
4. skewness_integrated_profile
5. mean_dm_snr_curve
6. std_deviation_dm_snr_curve
7. kurtosis_dm_snr_curve
8. skewness_dm_snr_curve
9. target_class

In [33]:
```
data.head()
```

Out[33]:

	Mean of the integrated profile	Standard deviation of the integrated profile	Excess kurtosis of the integrated profile	Skewness of the integrated profile	Mean of the DM-SNR curve	Standard deviation of the DM-SNR curve	Excess kurtosis of the DM-SNR curve	Skewness of the DM-SNR curve	target_class
0	140.562500	55.683782	-0.234571	-0.699648	3.199833	19.110426	7.975532	74.242225	0
1	102.507812	58.882430	0.465318	-0.515088	1.677258	14.860146	10.576487	127.393580	0
2	103.015625	39.341649	0.323328	1.051164	3.121237	21.744669	7.735822	63.171909	0
3	136.750000	57.178449	-0.068415	-0.636238	3.642977	20.959280	6.896499	53.593661	0
4	88.726562	40.672225	0.600866	1.123492	1.178930	11.468720	14.269573	252.567306	0

3. Analysing Data

Producing a heat-map for correlation between different features:

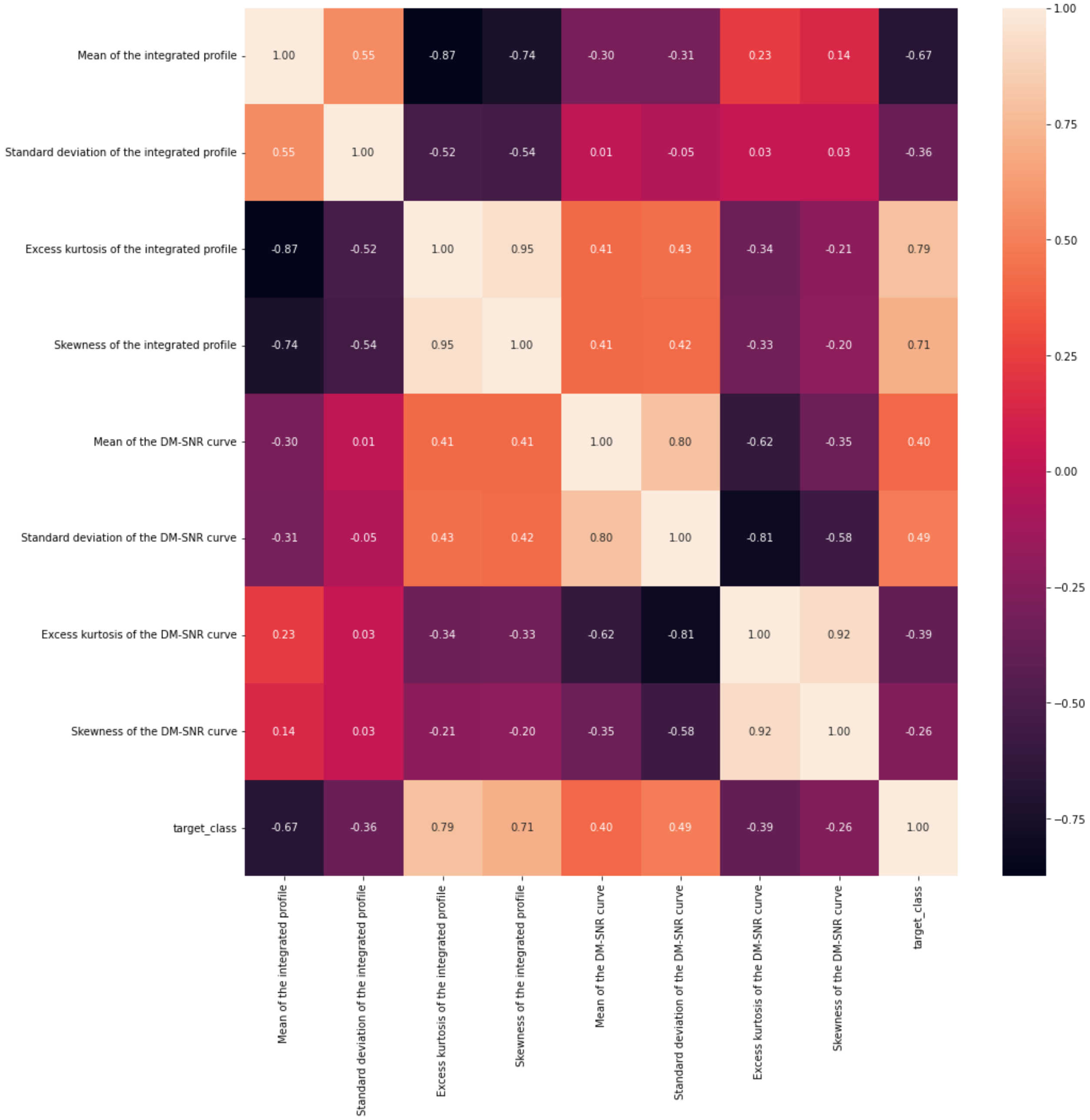
I have produced a heat map, which shows correlation between features.
There is a high positive correlation between following features:

- Excess kurtosis of the integrated profile - Skewness of the integrated profile (0.95)
- Mean of the DM-SNR curve - Standard deviation of the DM-SNR curve (0.80)
- Excess kurtosis of the DM-SNR curve - Skewness of the DM-SNR curve (0.92)

There is a high negative correlation between following features:

- Mean of the integrated profile - Excess kurtosis of the integrated profile (-0.87)
- Mean of the integrated profile - Skewness of the integrated profile (-0.74)
- Standard deviation of the DM-SNR curve - Excess kurtosis of the DM-SNR curve (-0.81)

In [30]:
```
f,ax=plt.subplots(figsize=(15,15))
sns.heatmap(data.corr(),annot=True,linecolor="blue",fmt=".2f",ax=ax)
plt.show()
```

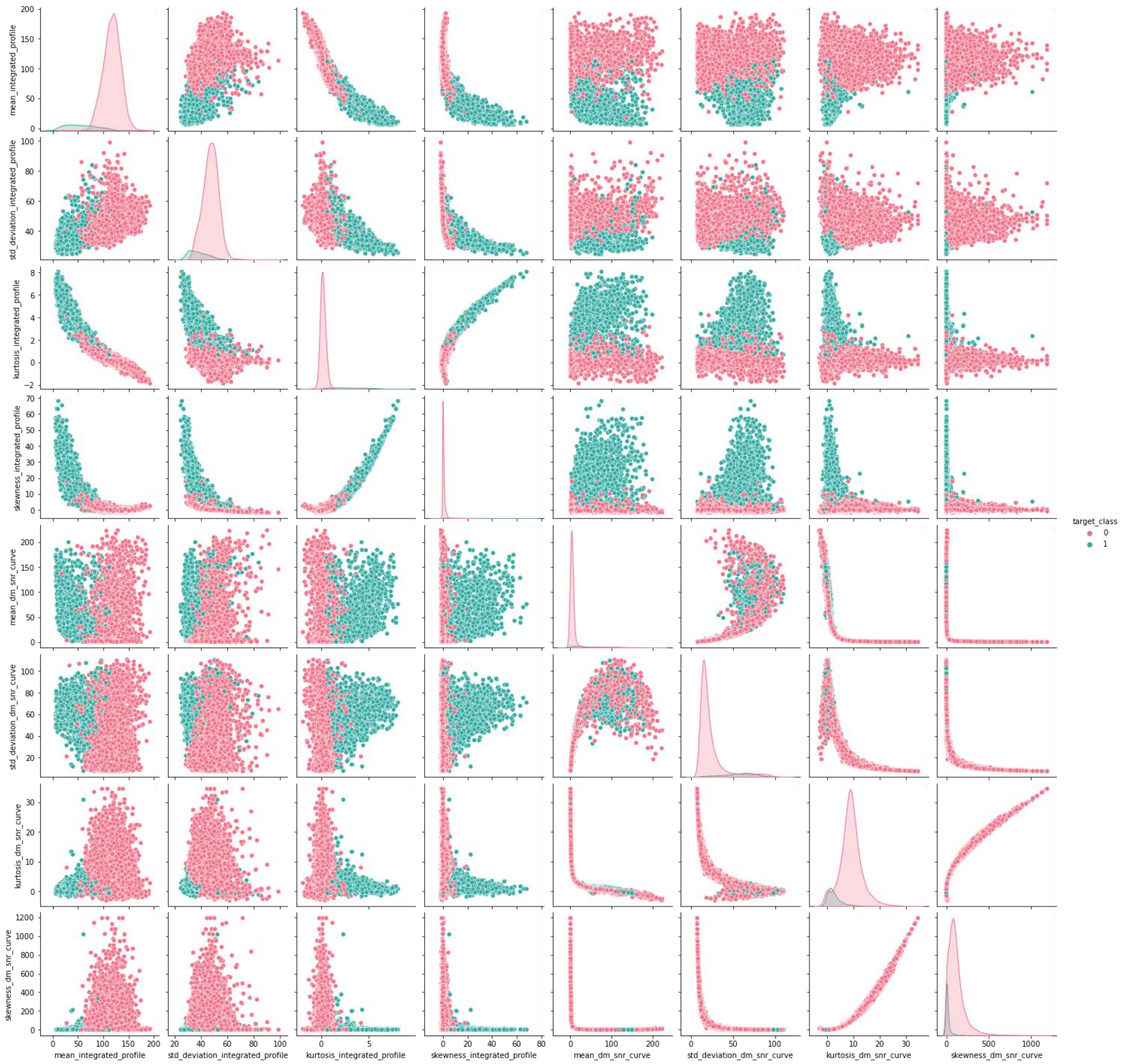

Mean of the integrated profile
Standard deviation of the integrated profile
Excess kurtosis of the integrated profile
Skewness of the integrated profile
Mean of the DM-SNR curve
Standard deviation of the DM-SNR curve
Excess kurtosis of the DM-SNR curve
Skewness of the DM-SNR curve
target_class
1.00 0.55 -0.87 -0.74 -0.30 -0.31 0.23 0.14 -0.67
0.55 1.00 -0.52 -0.54 0.01 -0.05 0.03 0.03 -0.36
-0.87 -0.52 1.00 0.95 0.41 0.43 -0.34 -0.21 0.79
-0.74 -0.54 0.95 1.00 0.41 0.42 -0.33 -0.20 0.71
-0.30 0.01 0.41 0.41 1.00 0.80 -0.62 -0.35 0.40
-0.31 -0.05 0.43 0.42 0.80 1.00 -0.81 -0.58 0.49
0.23 0.03 -0.34 -0.33 -0.62 -0.81 1.00 0.92 -0.39
0.14 0.03 -0.21 -0.20 -0.35 -0.58 0.92 1.00 -0.26
-0.67 -0.36 0.79 0.71 0.40 0.49 -0.39 -0.26 1.00
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75

Heat Map

Producing pair-plot to show correlation between features with classes:

The following pairplots show correlations between features with classes.

In [13]:
```
g = sns.pairplot(data, hue="target_class",palette="husl",diag_kind = "kde",kind = "scatter")
```



4. Data wrangling (pre-processing) and Testing Data:

The most important part of machine learning based application is processing of data. To prepare an accurate machine learning application data must be proper processed before training the actual model. Python machine learning frame work has been used for this data processing phase in the proposed model.

In [14]:
```
y = data["target_class"].values
x_data = data.drop(["target_class"],axis=1)
x = (x_data - np.min(x_data))/(np.max(x_data)-np.min(x_data))
```

In [15]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.3,random_state=1)
```

Next, we will set our predictor and target variable into x and y respectively. Then the whole data set is split into train and test data set using the train-test-split package from SciKit learning module. Next the model is built using another package from SciKit, LogisticRegression.

Logistic Regression

In [16]:
```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(x_train,y_train)
lr_prediction = lr.predict(x_test)
```

In [17]:
```
from sklearn.metrics import mean_squared_error
mse_lr=mean_squared_error(y_test,lr_prediction)

from sklearn.metrics import confusion_matrix,classification_report
cm_lr=confusion_matrix(y_test,lr_prediction)
cm_lr=pd.DataFrame(cm_lr)
cm_lr["total"]=cm_lr[0]+cm_lr[1]
cr_lr=classification_report(y_test,lr_prediction)
```

In [18]:
```
from sklearn.metrics import cohen_kappa_score
cks_lr= cohen_kappa_score(y_test, lr_prediction)
```

In [19]:
```
score_and_mse={"model":["logistic regression"],"Score":[lr.score(x_test,y_test)],"Cohen Kappa Score":[cks_lr],"MSE":[mse_lr]}
score_and_mse=pd.DataFrame(score_and_mse)
```

Classification Report for Logistic Regression:

In [20]:
```
print('Classification report for Logistic Regression:',cr_lr)
```

```
Classification report for Logistic Regression:               precision    recall  f1-score   support

           0       0.98      0.99      0.99      4880
           1       0.94      0.77      0.84       490

    accuracy                           0.97      5370
   macro avg       0.96      0.88      0.91      5370
weighted avg       0.97      0.97      0.97      5370
```

5. Finding Accuracy and Building a Confusion Matrix:

It is the most crucial step of our project. Here, we check the accuracy of the model, which highly depends on the pre-processing of the data. Since logistic regression is a supervised learning method, so the training of data is the most crucial step.

In [19]:
```
score_and_mse={"model":["logistic regression"],"Score":[lr.score(x_test,y_test)],"Cohen Kappa Score":[cks_lr],"MSE":[mse_lr]}
score_and_mse=pd.DataFrame(score_and_mse)
```

In [26]:
```
score_and_mse
```

Out[26]:

	model	Score	Cohen Kappa Score	MSE
0	logistic regression	0.974115	0.830036	0.025885

Accuracy (Score):

It is the ratio of number of correct predictions to the total number of input samples.

Accuracy= Number of Correct Predictions/ Total Number of Predictions Made

We got an accuracy of 97.41%

Cohen Kappa Score:

Kappa is similar to Accuracy score, but it considers the accuracy that would have happened anyway through random predictions.

It is a measure of how well the classifier actually performs. In other words, if there is a big difference between accuracy and null error rate, a model will have a high Kappa score.

Cohen Kappa only serves to make comparisons between two classifiers, if there are more than two classifiers, Fleiss's Kappa is used.

Kappa = (Observed Accuracy - Expected Accuracy) / (1 - Expected Accuracy)

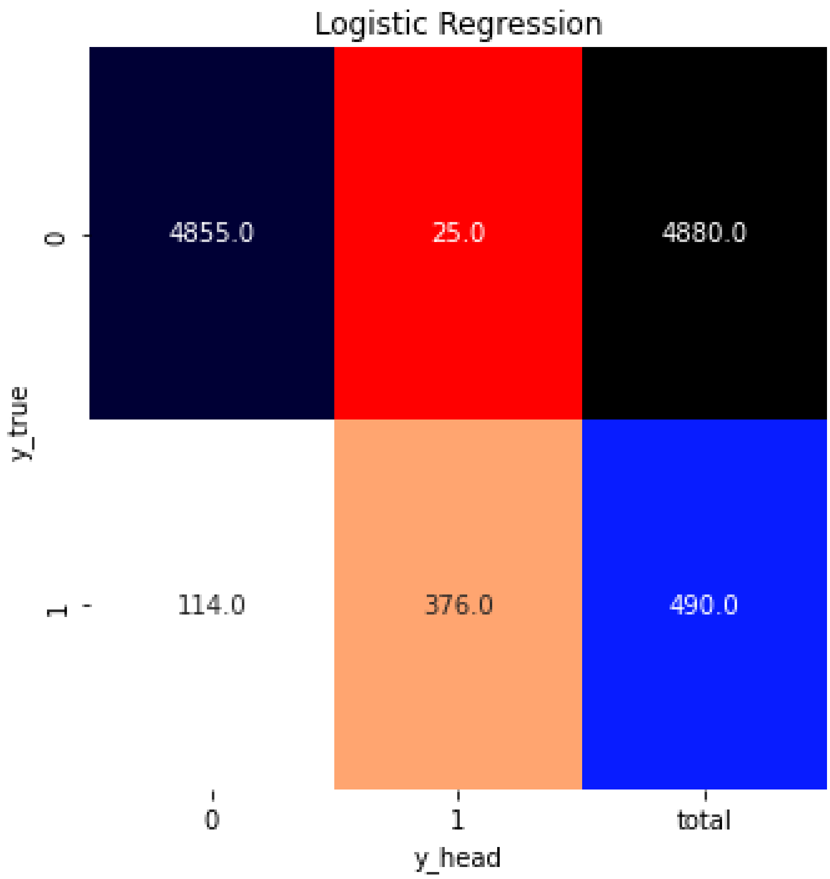
Mean Squared Error (MSE):

Mean Squared Error (MSE) is quite similar to Mean Absolute Error, the only difference being that MSE takes the average of the square of the difference between the original values and the predicted values.

❖ Confusion Matrix:

- A confusion matrix is a summary of prediction results on a classification problem.
- Positive (P): Observation is positive (for example: is a Pulse Star).
- Negative (N): Observation is not positive (for example: is not a Pulse Star).
- True Positive (TP): Observation is positive, and is predicted to be positive.
- False Negative (FN): Observation is positive, but is predicted negative.
- True Negative (TN): Observation is negative, and is predicted to be negative.
- False Positive (FP): Observation is negative, but is predicted positive.

In [25]:
```
f, axes = plt.subplots(2, 3,figsize=(18,12))
g1 = sns.heatmap(cm_lr,annot=True,fmt=".1f",cmap="flag",cbar=False,ax=axes[0,0])
g1.set_ylabel('y_true')
g1.set_xlabel('y_head')
g1.set_title("Logistic Regression")
```
Out[25]: `Text(0.5, 1.0, 'Logistic Regression')`



Confusion Matrix

Conclusion:

For such a huge Dataset, we found our accuracy quite significant.

```
In [28]: data
```

Out[28]:

	Mean of the integrated profile	Standard deviation of the integrated profile	Excess kurtosis of the integrated profile	Skewness of the integrated profile	Mean of the DM-SNR curve	Standard deviation of the DM-SNR curve	Excess kurtosis of the DM-SNR curve	Skewness of the DM-SNR curve	target_class
0	140.562500	55.683782	-0.234571	-0.699648	3.199833	19.110426	7.975532	74.242225	0
1	102.507812	58.882430	0.465318	-0.515088	1.677258	14.860146	10.576487	127.393580	0
2	103.015625	39.341649	0.323328	1.051164	3.121237	21.744669	7.735822	63.171909	0
3	136.750000	57.178449	-0.068415	-0.636238	3.642977	20.959280	6.896499	53.593661	0
4	88.726562	40.672225	0.600866	1.123492	1.178930	11.468720	14.269573	252.567306	0
...	...	...	...	...	...	...	...	...	...
17893	136.429688	59.847421	-0.187846	-0.738123	1.296823	12.166062	15.450260	285.931022	0
17894	122.554688	49.485605	0.127978	0.323061	16.409699	44.626893	2.945244	8.297092	0
17895	119.335938	59.935939	0.159363	-0.743025	21.430602	58.872000	2.499517	4.595173	0
17896	114.507812	53.902400	0.201161	-0.024789	1.946488	13.381731	10.007967	134.238910	0
17897	57.062500	85.797340	1.406391	0.089520	188.306020	64.712562	-1.597527	1.429475	0

17898 rows × 9 columns

References:

- ❖ https://en.wikipedia.org/wiki/Pulsar
- ❖ Kaggel Website
- ❖ GitHub website
- ❖ https://en.wikipedia.org/wiki/Logistic_regression

[I am uploading the whole program in my GitHub profile. It is accessible from here!]