Data Analytics Lab: Assignment-6
A Mathematical Essay on Support Vector Machine

Arjav Singh
Metallurgical and Materials Engineering
Indian Institute of Technology Madras
Chennai, India
mm20b007@smail.iitm.ac.in

***Abstract*—This study is a mathematical exposition of the Support Vector Machine method, which is applied to a dataset containing different start attributes. The primary objective is to differentiate pulsar stars from non-pulsar stars by utilizing different metrics derived from their integrated pulse profiles (folded profiles). The essay aims to assess the efficacy of SVMs in this classification task and compare the performance of different kernels. Additionally, it investigates whether specific features exhibit distinctive characteristics crucial for accurate classification between the two-star classes.**

***Index Terms*—Introduction, Support Vector Machine, Data & Problem, Conclusion**

I. Introduction

Pulsars are rotating neutron stars observed to have pulses of radiation at regular intervals that typically range from milliseconds to seconds. Pulsars have very strong magnetic fields which funnel jets of particles out along the two magnetic poles. These accelerated particles produce very powerful beams of light.

This study focuses on a comprehensive empirical classification of a pulsar from a normal star based on its features, including the *Mean of the integrated profile*, *Excess kurtosis of the integrated profile*, *Skewness of the integrated profile*, *Mean of the DM-SNR curve*, *Excess kurtosis of the DM-SNR curve*, *Skewness of the DM-SNR curve*. Support Vector Machine Classification Machine learning technique is used to achieve the goal. In SVM, the data points are first represented in an n-dimensional space. The algorithm then uses statistical approaches to find the best line that separates the various classes present in the data.

The research methodology initiates with the acquisition, refinement, and preprocessing of the raw data. An exploratory data analysis follows this to gain a deeper understanding of the dataset's inherent features. Subsequently, statistical models are crafted, along with the generation of visual aids to offer both quantitative and visual support for the observed associations. The subsequent section furnishes an exposition and discourse on the insights and revelations extracted from the data analysis and the models that have been generated. It emphasizes the significant discoveries, recurring patterns, and emerging trends that have surfaced during the course of the study.

The concluding section summarizes the key highlights and significant features of the research. Potential avenues for further investigation are outlined, suggesting areas where future research could expand upon the findings. A contribution is made to a deeper understanding of pulsars and how to identify them, with valuable insights for their detection.

II. Support Vector Machine

The Support Vector Machine (SVM) is a robust supervised algorithm ideally suited for handling complex datasets. SVM can be employed for regression and classification tasks, although it typically excels in classification problems. Despite being developed in the 1990s, SVM remains a popular choice, known for its high-performance capabilities even with minimal parameter tuning.

A. Types and Features of SVM

1) **Linear SVM**: When the data is perfectly linearly separable, we can only use Linear SVM. Perfectly linearly separable means that the data points can be classified into 2 classes by using a single straight line(if 2D).
2) **Non-Linear SVM**: When the data is not linearly separable, then we can use Non-Linear SVM, which means when the data points cannot be separated into 2 classes by using a straight line (if 2D) then we use some advanced techniques like kernel tricks to classify them. In most real-world applications, we do not find linearly separable data points; hence we use kernel tricks to solve them.

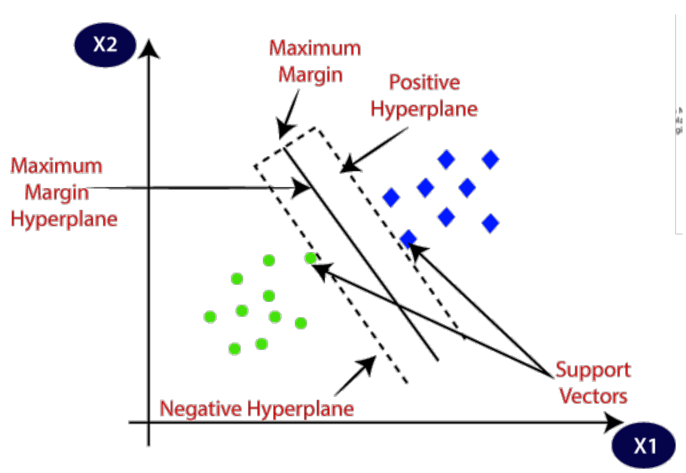


Fig. 1. General Description of Support Vector Machine

An SVM has two major components(Figure 1), which are

1) **Support Vectors:**: These are the points closest to the hyperplane. A separating line will be defined with the help of these data points.
2) **Margin:** It is the distance between the hyperplane and the observations closest to the hyperplane (support vectors). In SVM large margin is considered a good margin. There are two types of margins hard margin and soft margin.

B. Working of SVM

In SVMs, we mainly aim to select a hyperplane with the maximum possible margin between support vectors in the given dataset. SVM searches for the maximum margin hyperplane in the following 2-step process –

1) Generate hyperplanes that segregate the classes in the best possible way. Many hyperplanes might classify the data. We should look for the best hyperplane representing the largest separation, or margin, between the two classes.
2) So, we choose the hyperplane so that the distance from it to the support vectors on each side is maximized. If such a hyperplane exists, it is known as the maximum margin hyperplane, and the linear classifier it defines is known as a maximum margin classifier.

Figure 2 illustrates the concept of maximum margin and maximum margin hyperplane in a clear manner.

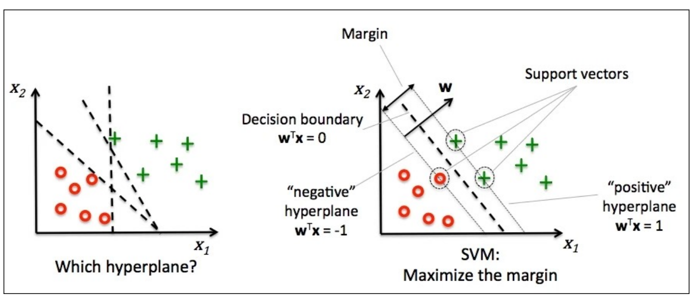


Fig. 2. Working of SVM

C. Problem with dispersed datasets

Sometimes, the sample data points are so dispersed that it is impossible to separate them using a linear hyperplane. In such a situation, SVMs use a kernel trick to transform the input space to a higher dimensional space, as shown in Figure 3. It uses a mapping function to transform the 2-D input space into the 3-D input space. Now, we can easily segregate the data points using linear separation.

D. Kernel Methods for SVM

In practice, SVM algorithm is implemented using a kernel. It uses a technique called the kernel trick. Simply put, a kernel is just a function that maps the data to a higher dimension where data is separable. A kernel transforms a low-dimensional input data space into a higher-dimensional

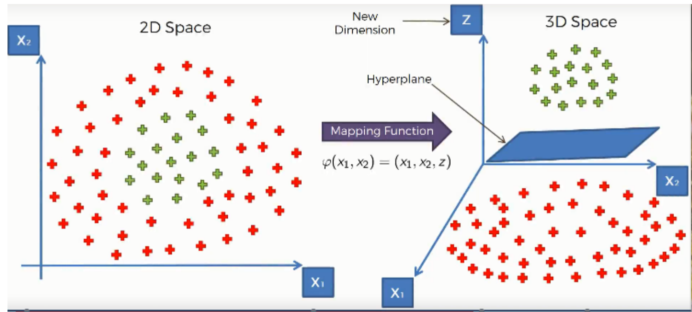


Fig. 3. Working of SVM on non-linear data

space. So, it converts non-linear separable problems to linear separable problems by adding more dimensions to it. Thus, the kernel trick helps us to build a more accurate classifier. Hence, it is useful in non-linear separation problems. We can define a kernel function as follows-

$$K\left(\overline{x}\right) = \begin{matrix} 1 & \text{if} \left\|\overline{x}\right\| \leq 1 \\ 0 & \text{otherwise} \end{matrix}$$

Fig. 4. Kernel function

In the context of SVMs, there are 4 popular kernels – Linear kernel, Polynomial kernel, Radial Basis Function (RBF) kernel (also called Gaussian kernel), and Sigmoid kernel. These are described below -

1) **Linear kernel:** In linear kernel, the kernel function takes the form of a linear function as follows linear kernel: K(x_i, x_j) = $x_i^T x_j$.
Linear kernel is used when the data is linearly separable. It means that data can be separated using a single line. It is one of the most common kernels to be used. It is mostly used when there are large number of features in a dataset. Linear kernel is often used for text classification purposes. Training with a linear kernel is usually faster because we only need to optimize the C regularization parameter. When training with other kernels, we also need to optimize the parameter. So, performing a grid search will usually take more time. The linear kernel can be visualized in Figure 5.
2) **Polynomial kernel:** Polynomial kernel represents the similarity of vectors (training samples) in a feature space over polynomials of the original variables. The polynomial kernel looks not only at the given features of input samples to determine their similarity but also at combinations of the input samples.
For d-degree polynomials, the polynomial kernel is defined as follows: Polynomial kernel:
$$K(\mathbf{x}_i, \mathbf{x}_j) = (\gamma \mathbf{x}_i^T \mathbf{x}_j + r)^d, \gamma > 0$$

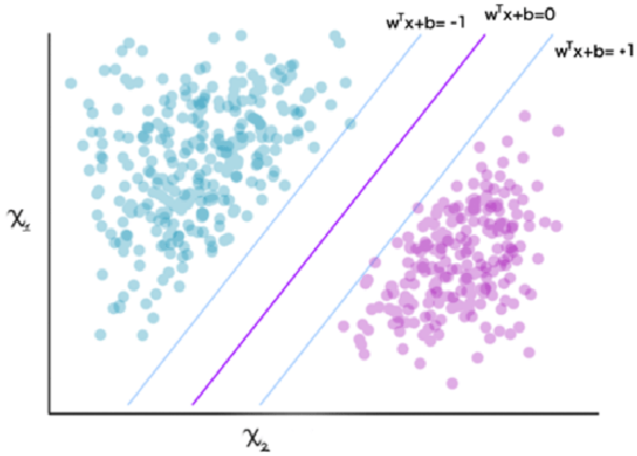


Fig. 5. Linear Kernel function

The polynomial kernel is very popular in Natural Language Processing. The most common degree is $d = 2$ (quadratic) since larger degrees tend to overfit NLP problems. It can be visualized in Figure 6.

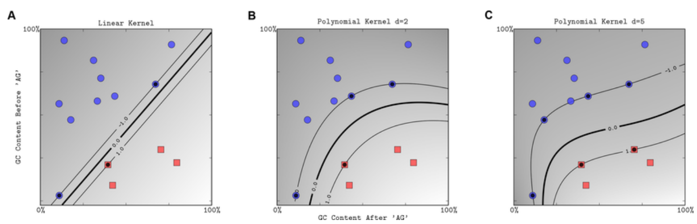


Fig. 6. Polynomial Kernel function

3) **Radial basis function kernel:** Radial basis function kernel is a general purpose kernel. It is used when we have no prior knowledge about the data. The RBF kernel on two samples, x and y, is defined by the following equation –

$$k(x, y) = \exp\left(-\frac{\|x - y\|^2}{2\sigma^2}\right)$$

Fig. 7. Radial Basis function Kernel Equation

4) **Sigmoid Function Kernel:** The sigmoid kernel originates in neural networks and can be used as a proxy for neural networks. The following equation gives the sigmoid kernel: Sigmoid kernel:

$$k(\mathbf{x}, \mathbf{y}) = \tanh(\alpha \mathbf{x}^T \mathbf{y} + c)$$

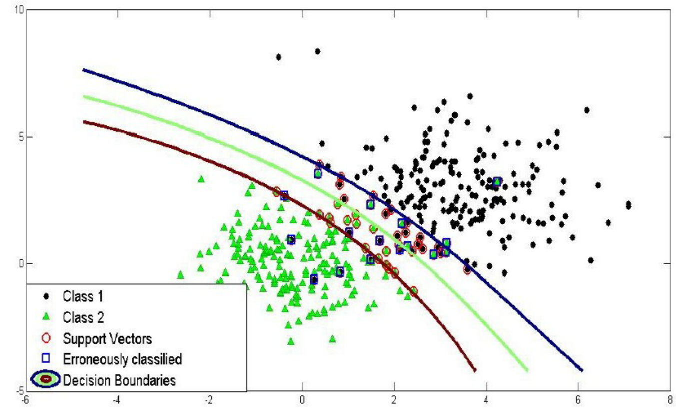


Fig. 8. Classification using radial basis function kernel

E. Metrics for model evaluation

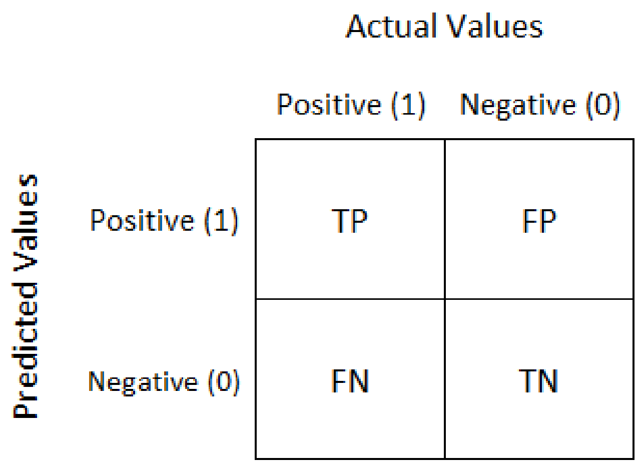


Fig. 9. Confusion Matrix.

1) **Confusion Matrix**: It is used to summarize the performance of a classification algorithm on a set of test data for which the true values are previously known. Sometimes it is also called an error matrix. Terminologies of the Confusion matrix (Figure 1) are:
   - **True Positive**: TP means the model predicted yes, and the actual answer is also yes.
   - **True negative**: TN means the model predicted no, and the actual answer is also no.
   - **False positive**: FP means the model predicted yes, but the actual answer is no.
   - **False negative**: FN means the model predicted no, but the actual answer is yes.

   The rates calculated using the Confusion Matrix are:

   a) **Accuracy**: (TP+TN/Total) tells about overall how classifier Is correct.

b) **True positive rate**: TP/(actual yes) it says about how much time yes is predicted correctly. It is also called “sensitivity” or “recall.”

c) **False positive rate**: FP/(actual number) says how much time yes is predicted when the actual answer is no.

d) **True negative rate**: TN/(actual number) says how much time no is predicted correctly, and the actual answer is also no. It is also known as “specificity.”

e) **Misclassification rate**: (FP+FN)/(Total) It is also known as the error rate and tells about how often our model is wrong.

f) **Precision**: (TP/ (predicted yes)) If it predicts yes, then how often is it correct.

g) **Prevalence**: (actual yes /total) how often yes condition actually occurs.

h) **F1-score**: f1 score is defined as the weighted harmonic mean of precision and recall. The best achievable F1 score is 1.0, while the worst is 0.0. The F1 score serves as the harmonic mean of precision and recall. Consequently, the F1-score consistently yields lower values than accuracy measures since it incorporates precision and recall in its computation. When evaluating classifier models, it is advisable to employ the weighted average of the F1 score instead of relying solely on global accuracy.

2) **ROC curve (Receiver Operating Characteristic)**: The Receiver Operating Characteristic (ROC) curve is a useful tool for assessing a model’s performance by examining the trade-offs between its True Positive (TP) rate, also known as sensitivity, and its False Negative (FN) rate, which is the complement of specificity. This curve visually represents these two parameters.
The Area Under the Curve (AUC) metric to summarize the ROC curve concisely. The AUC quantifies the area under the ROC curve. In simpler terms, it measures how well the model can distinguish between positive and negative cases. A higher AUC indicates better classifier performance.
In essence, AUC categorizes model performance as follows:
   - If AUC = 1, the classifier correctly distinguishes between all the Positive and Negative class points.
   - If 0.5¡ AUC ¡ 1, the classifier will distinguish the positive class value from the negative one because it finds more TP and TN than FP and FN.
   - If AUC = 0.5, the classifier cannot distinguish between positive and negative values.
   - If AUC =0, the classifier predicts all positive as negative and negative as positive.

III. Problem

We have been tasked to analyze various attributes of stars, such as their Mean of the integrated profile, Excess kurtosis of the integrated profile, Skewness of the integrated profile, Mean of the DM-SNR curve, Excess kurtosis of the DM- SNR curve, Skewness of the DM-SNR curve. The goal is to identify which of them are pulsars.

A. Exploratory Data Analysis and Feature Generation

The data is initially read into a pandas data frame. A total of 12528 data points are observed, with 9 columns encompassing various car-related features. When the distributions of the target variable are visualized, a multi-class imbalanced dataset problem is evident. Around 90.8% of the total stars are classified as not pulsars, and only 9.2% are classified as pulsars (as shown in Figure 11). It is observed that 8 out of 9 features are continuous and are numerical. The target feature is categorical in nature and has labels 0 and 1. The aim is to predict this target feature.

Further analysis of the data implies that three features, namely *Excess kurtosis of the integrated profile, Standard deviation of the DM-SNR curve,* and *Skewness of the DM-SNR curve* have missing values.

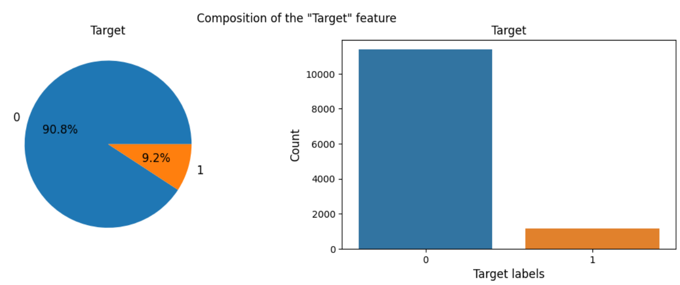


Fig. 10. Distribution of Target Variable

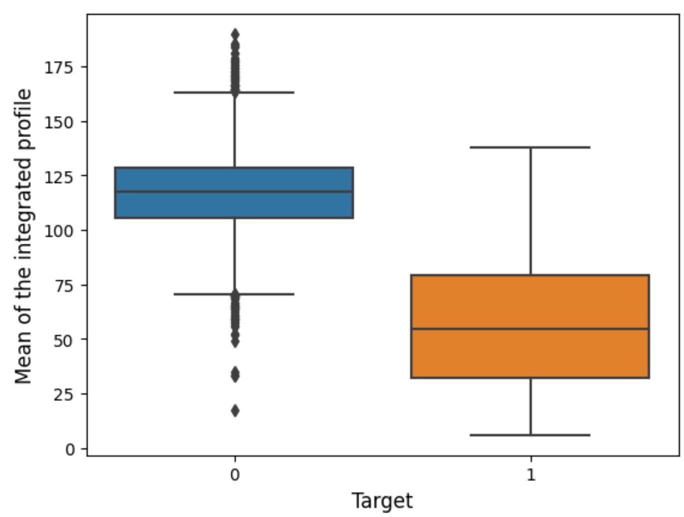


Fig. 11. Mean of the integrated profile vs Target

Univariate analysis is initiated by generating boxplots for each of the eight features, employing the seaborn library, with the target column as the hue. It is observed that there is an opposite trend for the same values of integrated profile and DM-SNR. Correlation is then checked, and Excess kurtosis of

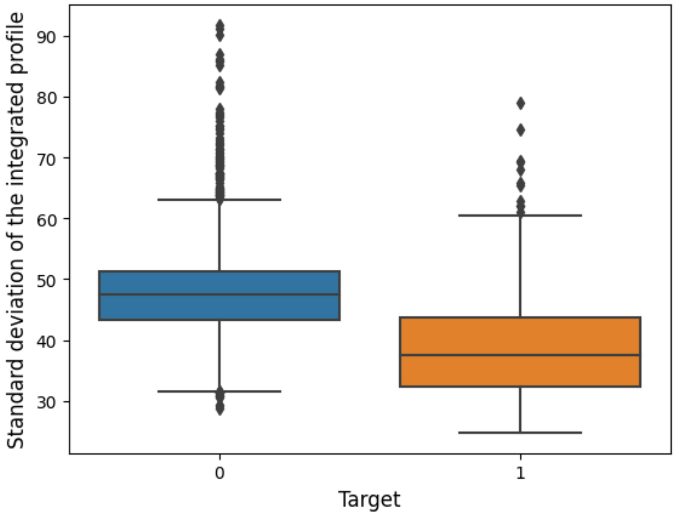


Fig. 12. Standard deviation of the integrated profile vs Target

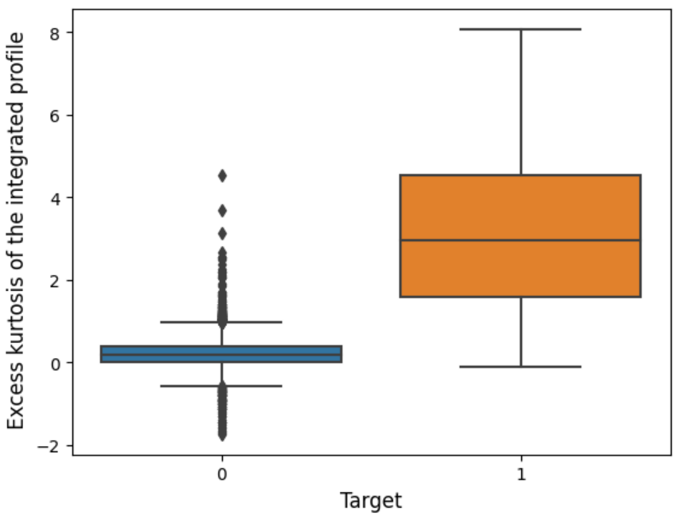


Fig. 13. Excess kurtosis of the integrated profile vs Target

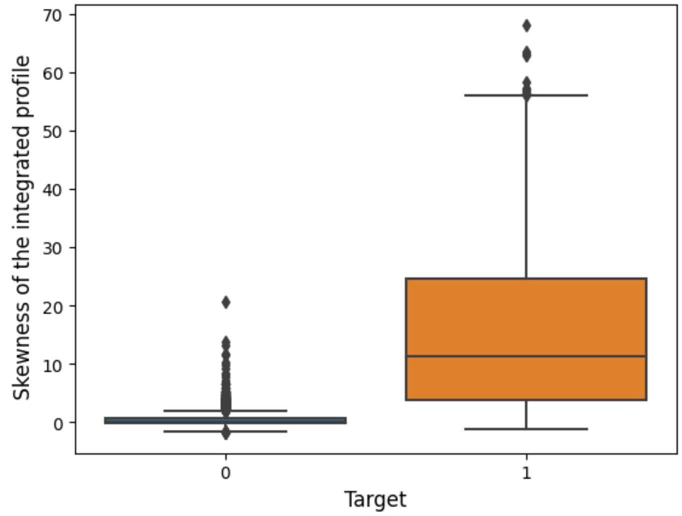


Fig. 14. Skewness of the integrated profile vs Target

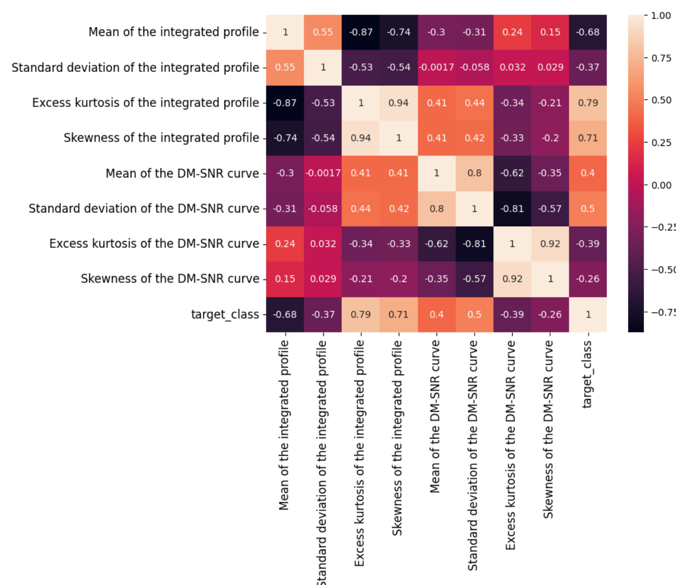


Fig. 15. Correlation heatmap

the integrated profile, Skewness of the integrated profile, and Standard deviation of the DM-SNR curve have a high positive correlation (greater than 0.5) with the target class feature. In contrast, apart from the Mean of the DM-SNR curve, all remaining features negatively associate with the target class feature.

B. Post-Processing and Feature Selection

Since the dataset comprises missing values, it is necessary to handle it properly. There are many methods of imputation, for the given problem, I have used two methods of imputation:

1) Standard imputation: The missing values were replaced with median and mean for the respective features.
2) Iterative imputation: This method first fits the data and generates a function that predicts the missing values. It is visible in Figure 17 that this is a better version of the imputation for the given dataset type.

Ultimately, the data is divided using an 80/20 split, resulting in a final dataset with 10022 examples in the training set and 2506 in the cross-validation set. We then use the Standard scaler to scale our train and validation values.

C. SVM Modelling

Modeling is initiated using the default hyperparameters provided by the SVC library in scikit-learn, where the defaults are set as follows: $C = 1.0$, $kernel = rbf$, and $gamma = auto$, among other parameters. Initial observations with default hyperparameters show that an accuracy score of 0.9816, and a

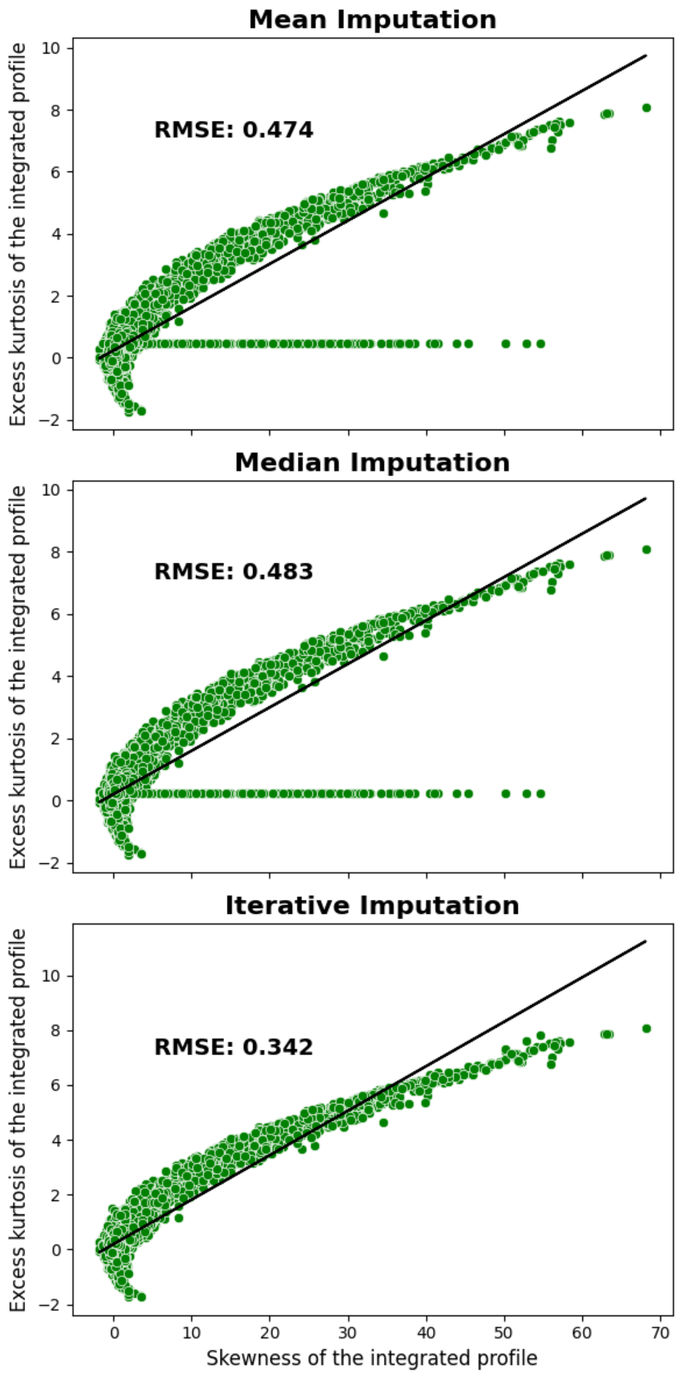


Fig. 16. Simple and Iterative imputation

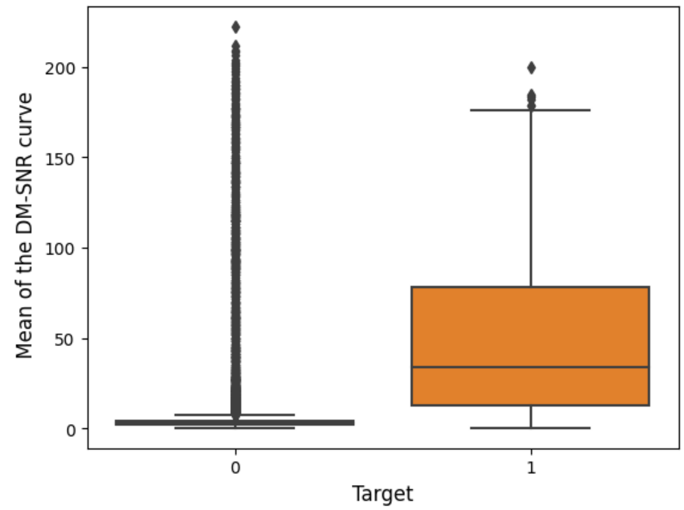


Fig. 17. Mean of the DM-SNR curve vs Target

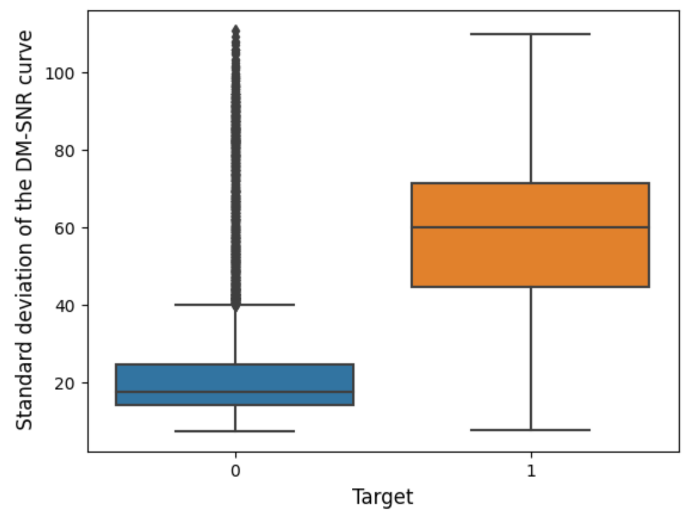


Fig. 18. Standard deviation of the DM-SNR curve vs Target

precision of 0.9594 are achieved by the model. These metrics imply the good performance of the model.

However, it is essential to note that our dataset is imbalanced. In this context, accuracy alone is an inadequate measure for assessing predictive performance. Alternative metrics that offer better insights into model selection must be explored. In particular, attention is turned to the F1 score, which is more informative when dealing with imbalanced datasets. It is found that it is found that the model achieved an F1 score of 0.8915 and an ROC AUC score of 0.9145.

To further enhance model performance, hyperparameter tuning is performed. Grid Search is employed to explore a predefined hyperparameter space, which includes testing various kernels and a range of C values from 1 to 10. Additionally, experimentation is done with the degree for the linear kernel and class weights are set as "balanced." Other kernel methods were also experimented with varying values

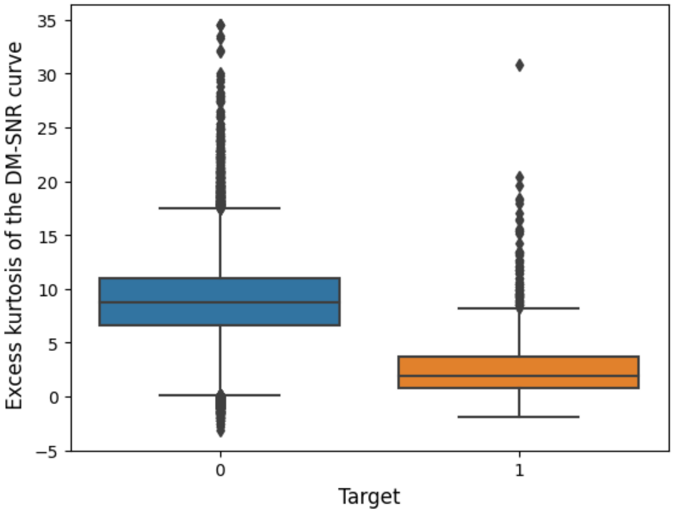


Fig. 19. Excess kurtosis of the DM-SNR curve vs Target

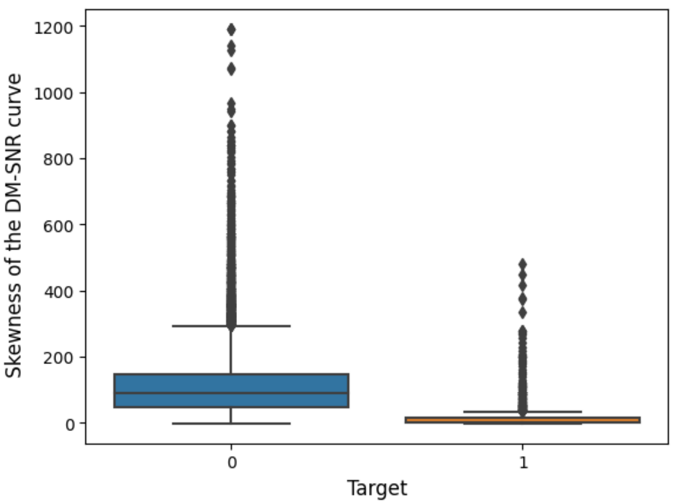


Fig. 20. Skewness of the DM-SNR curve vs Target

of the gamma hyperparameter, for polynomial kernel, different degrees were experimented with, ranging from 2 to 5. Utilizing a 2-fold cross-validation technique.

The best model is identified, with the following parameters $'C' : 10,' kernel' :' linear'$ exhibiting an F1 score of 0.9019, an accuracy of 0.9832, a precision of 0.9602, and an ROC AUC score of 0.9234. This represents an improvement over the initial F1 score of 0.8915.

IV. Conclusion

Having conducted a comprehensive analysis of the Support Vector methods with various kernels, an improvement of 0.01 in the F1 score was observed using GridSearchCV. Consequently, GridSearchCV serves the purpose of identifying the parameters that will enhance the performance of this specific model. The dataset contains outliers, and as the value of C was increased to reduce the influence of outliers, accuracy

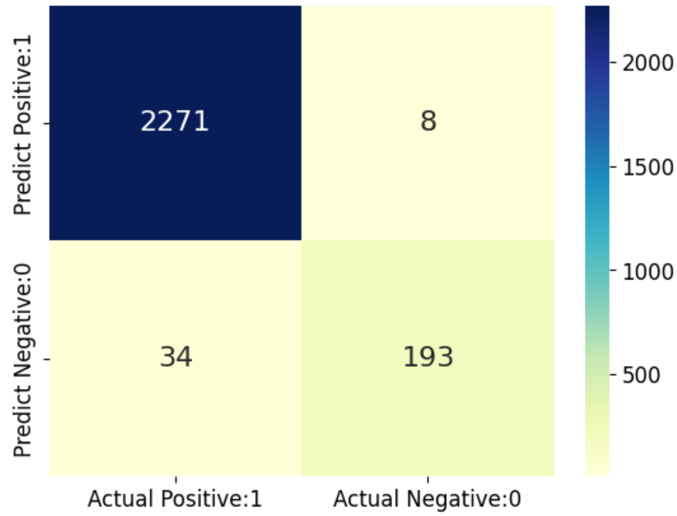


Fig. 21. Confusion Matrix for SVM after grid search

improved. This effect was consistent across different kernel types.

The ROC AUC of the model is very close to 1, suggesting that the classifier excels in classifying pulsar stars. Additionally, the precision and recall values are commendable, standing at 0.9602 and 0.985, respectively. The true positive rate is 0.985, while the false positive rate is 0.191.

In conclusion, SVMs with the linear kernel demonstrate the capability to fit the training data effectively, as anticipated based on the multivariate analysis, which revealed that most features partition the target classes into distinct and easily separable regions. Future possibilities for improvement include exploring additional features that may provide better insights into the target variable. Additionally, addressing feature correlation by eliminating specific features or creating hybrid features could be explored. Given the highly imbalanced data, implementing upsampling and downsampling techniques is another avenue worth considering."

For future work, further avenues of growth could involve exploring additional features that might better explain the target variable.

References

[1] "Everything About SVM Classification: Above and Beyond," Online. Available: https://towardsdatascience.com/everything-about-svm-classification-above-and-beyond-cc665bfd993e.

[2] P. 111, "SVM Classifier Tutorial," Kaggle, Available: https://www.kaggle.com/code/prashant111/svm-classifier-tutorial/notebook.

[3] "Support Vector Machines (SVM): A Complete Guide for Beginners," Analytics Vidhya. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/.

[4] "AUC-ROC Curve & Confusion Matrix Explained in Detail." [Online]. Available: https://www.kaggle.com/code/vithal2311/auc-roc-curve-confusion-matrix-explained-in-detail.

[5] Analytics Vidhya. "K-Fold Cross-Validation Technique and Its Essentials." [Online]. Available: https://www.analyticsvidhya.com/blog/2022/02/k-fold-cross-validation-technique-and-its-essentials/.

MM20B007 DAL Assignment 6

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import SimpleImputer, IterativeImputer
from sklearn.metrics import mean_squared_error, precision_score,
accuracy_score, f1_score, roc_auc_score
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report, roc_curve

train_data_path = '/content/drive/MyDrive/sem 7/EE5708/Assignment
6/pulsar_data_train.xlsx'
test_data_path = '/content/drive/MyDrive/sem 7/EE5708/Assignment
6/pulsar_data_test.xlsx'

train_data = pd.read_excel(train_data_path)

train_data.head()
```

```
    Mean of the integrated profile  \
0                       121.156250
1                        76.968750
2                       130.585938
3                       156.398438
4                        84.804688

    Standard deviation of the integrated profile  \
0                                       48.372971
1                                       36.175557
2                                       53.229534
3                                       48.865942
4                                       36.117659

    Excess kurtosis of the integrated profile  \
0                                    0.375485
1                                    0.712898
2                                    0.133408
3                                   -0.215989
4                                    0.825013

    Skewness of the integrated profile   Mean of the DM-SNR curve  \
```

```
0                              -0.013165                    3.168896
1                               3.388719                    2.399666
2                              -0.297242                    2.743311
3                              -0.171294                   17.471572
4                               3.274125                    2.790134

    Standard deviation of the DM-SNR curve  \
0                                18.399367
1                                17.570997
2                                22.362553
3                                      NaN
4                                20.618009

    Excess kurtosis of the DM-SNR curve   Skewness of the DM-SNR curve
\
0                              7.449874                      65.159298

1                              9.414652                     102.722975

2                              8.508364                      74.031324

3                              2.958066                       7.197842

4                              8.405008                      76.291128


   target_class
0             0
1             0
2             0
3             0
4             0
```

```
train_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12528 entries, 0 to 12527
Data columns (total 9 columns):
 #   Column                                         Non-Null Count
Dtype
---  ------                                         --------------
-----
 0    Mean of the integrated profile                12528 non-null
float64
 1    Standard deviation of the integrated profile  12528 non-null
float64
 2    Excess kurtosis of the integrated profile     10793 non-null
float64
 3    Skewness of the integrated profile            12528 non-null
float64
```

```
 4    Mean of the DM-SNR curve                      12528 non-null
float64
 5    Standard deviation of the DM-SNR curve        11350 non-null
float64
 6    Excess kurtosis of the DM-SNR curve           12528 non-null
float64
 7    Skewness of the DM-SNR curve                  11903 non-null
float64
 8   target_class                                   12528 non-null
int64
dtypes: float64(8), int64(1)
memory usage: 881.0 KB
```

```
train_data.describe()
```

```
        Mean of the integrated profile  \
count                     12528.000000
mean                        111.041841
std                          25.672828
min                           5.812500
25%                         100.871094
50%                         115.183594
75%                         127.109375
max                         189.734375

        Standard deviation of the integrated profile  \
count                                   12528.000000
mean                                       46.521437
std                                         6.801077
min                                        24.772042
25%                                        42.362222
50%                                        46.931022
75%                                        50.979103
max                                        91.808628

        Excess kurtosis of the integrated profile  \
count                                10793.000000
mean                                     0.478548
std                                      1.064708
min                                     -1.738021
25%                                      0.024652
50%                                      0.223678
75%                                      0.473125
max                                      8.069522

        Skewness of the integrated profile   Mean of the DM-SNR curve
\
count                         12528.000000               12528.000000

mean                              1.778431                  12.674758
```

```
std                                     6.208450                    29.613230

min                                    -1.791886                     0.213211

25%                                    -0.188142                     1.910535

50%                                     0.203317                     2.792642

75%                                     0.932374                     5.413253

max                                    68.101622                   222.421405


        Standard deviation of the DM-SNR curve  \
count                              11350.000000
mean                                  26.351318
std                                   19.610842
min                                    7.370432
25%                                   14.404353
50%                                   18.412402
75%                                   28.337418
max                                  110.642211

        Excess kurtosis of the DM-SNR curve   Skewness of the DM-SNR
curve  \
count                          12528.000000
11903.000000
mean                               8.333489
105.525779
std                                4.535783
107.399585
min                               -3.139270                          -
1.976976
25%                                5.803063
35.199899
50%                                8.451097
83.126301
75%                               10.727927
139.997850
max                               34.539844
1191.000837

       target_class
count  12528.000000
mean       0.092034
std        0.289085
min        0.000000
25%        0.000000
50%        0.000000
```

```
75%          0.000000
max          1.000000
```

```
for cols in list(train_data.columns):
  s = train_data[cols].isna().sum()
  print(f'No. of missing values in {cols} are {s}')
```

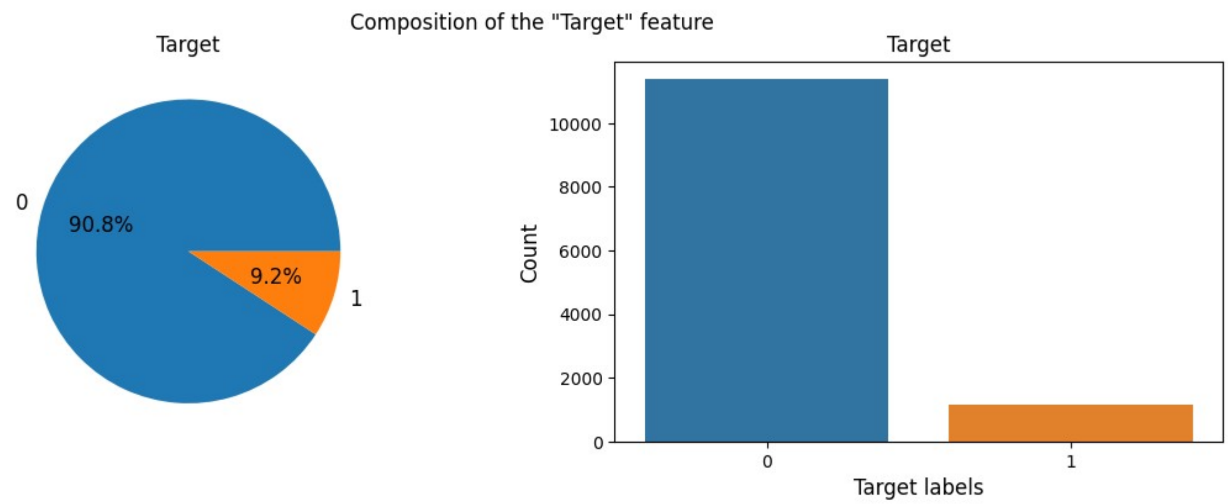
```
No. of missing values in  Mean of the integrated profile are 0
No. of missing values in  Standard deviation of the integrated profile
are 0
No. of missing values in  Excess kurtosis of the integrated profile
are 1735
No. of missing values in  Skewness of the integrated profile are 0
No. of missing values in  Mean of the DM-SNR curve are 0
No. of missing values in  Standard deviation of the DM-SNR curve are
1178
No. of missing values in  Excess kurtosis of the DM-SNR curve are 0
No. of missing values in  Skewness of the DM-SNR curve are 625
No. of missing values in target_class are 0
```

Observations

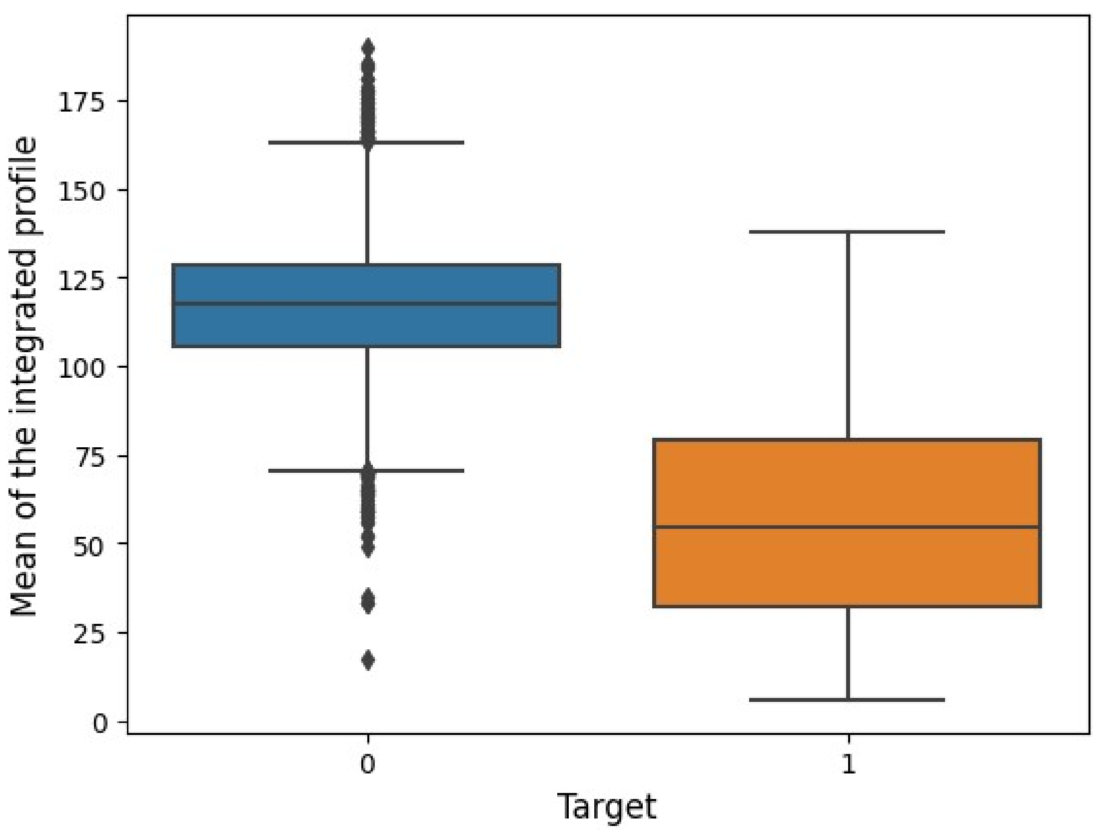
1. There are total of 12528 datapoints for each of 8 features and 1 target.
2. There are 3 features with missing data points
    - Excess kurtosis of the integrated profile - 1735
    - Standard deviation of the DM-SNR curve - 1178
    - Skewness of the DM-SNR curve - 625

```
df = train_data.copy()

fig, ax = plt.subplots(1, 2, figsize = (14, 4))
fig.suptitle('Composition of the "Target" feature')
df['target_class'].value_counts().plot.pie(ax = ax[0],autopct='%1.1f%
%',shadow=False, textprops={'fontsize': 12})
ax[0].set_title('Target')
ax[0].set_ylabel(None)
sns.countplot(x = 'target_class', data = df, ax=ax[1])
ax[1].set_title('Target')
ax[1].set_ylabel('Count', fontsize = 12)
ax[1].set_xlabel('Target labels', fontsize = 12)
```

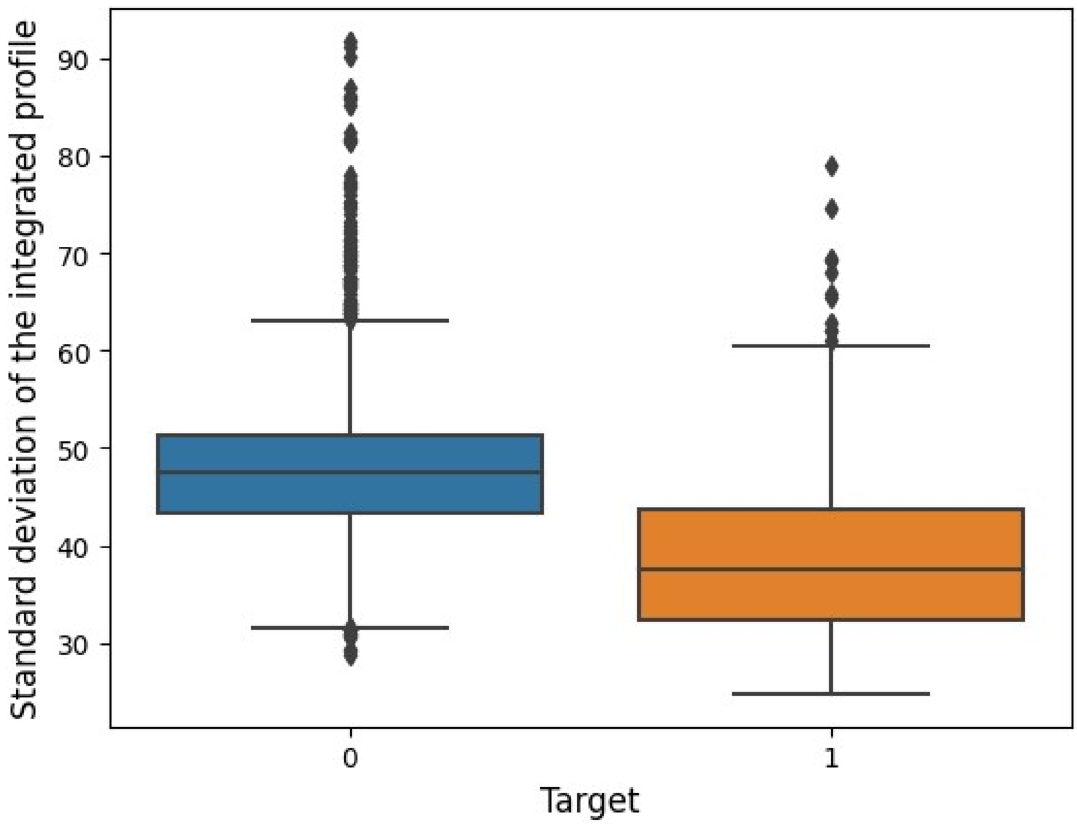
```
Text(0.5, 0, 'Target labels')
```



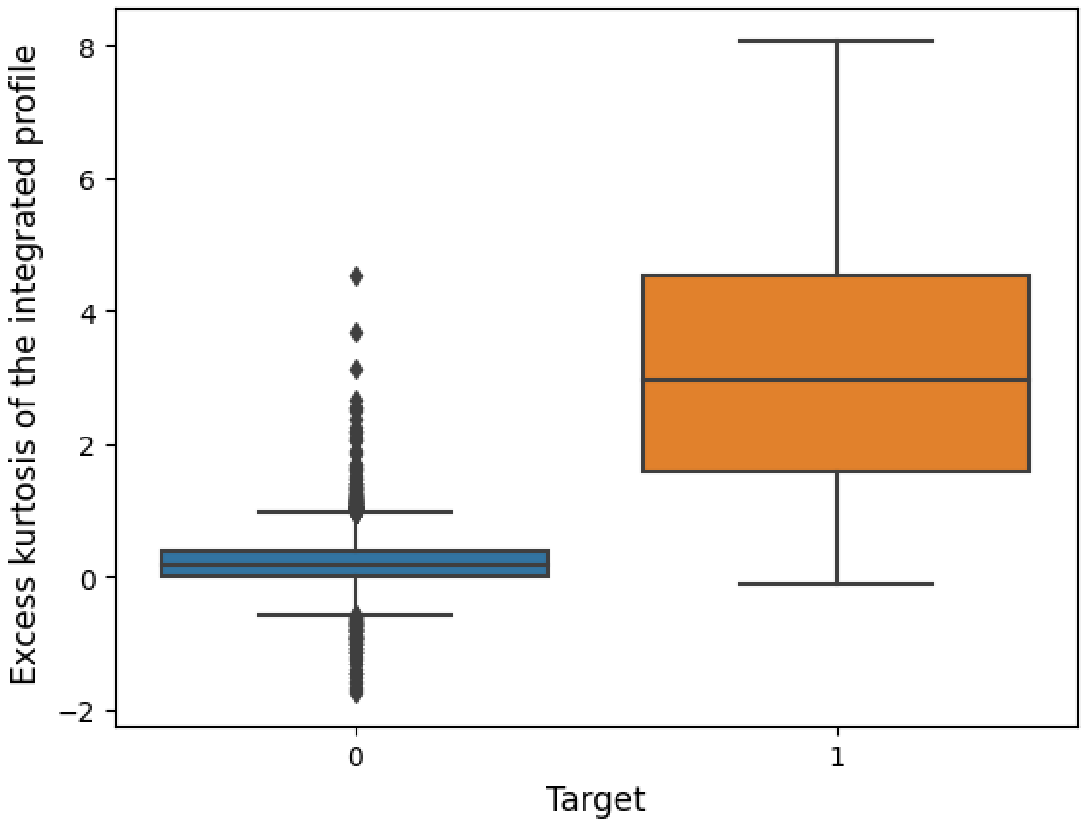

```
sns.boxplot(df, y = ' Mean of the integrated profile' ,x =
'target_class')
plt.ylabel('Mean of the integrated profile', fontsize = 12)
plt.xlabel('Target', fontsize = 12)

Text(0.5, 0, 'Target')
```



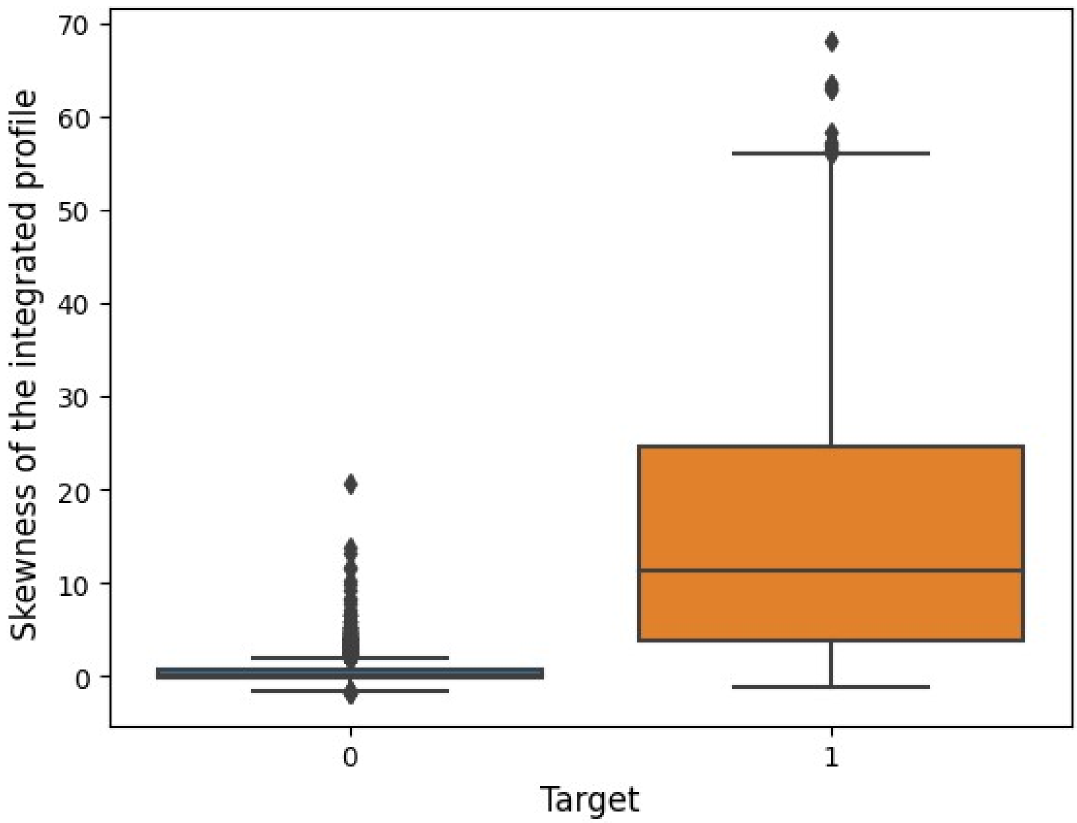
```
sns.boxplot(df, y = ' Standard deviation of the integrated profile' ,x
= 'target_class')
plt.ylabel('Standard deviation of the integrated profile', fontsize =
12)
plt.xlabel('Target', fontsize = 12)

Text(0.5, 0, 'Target')
```



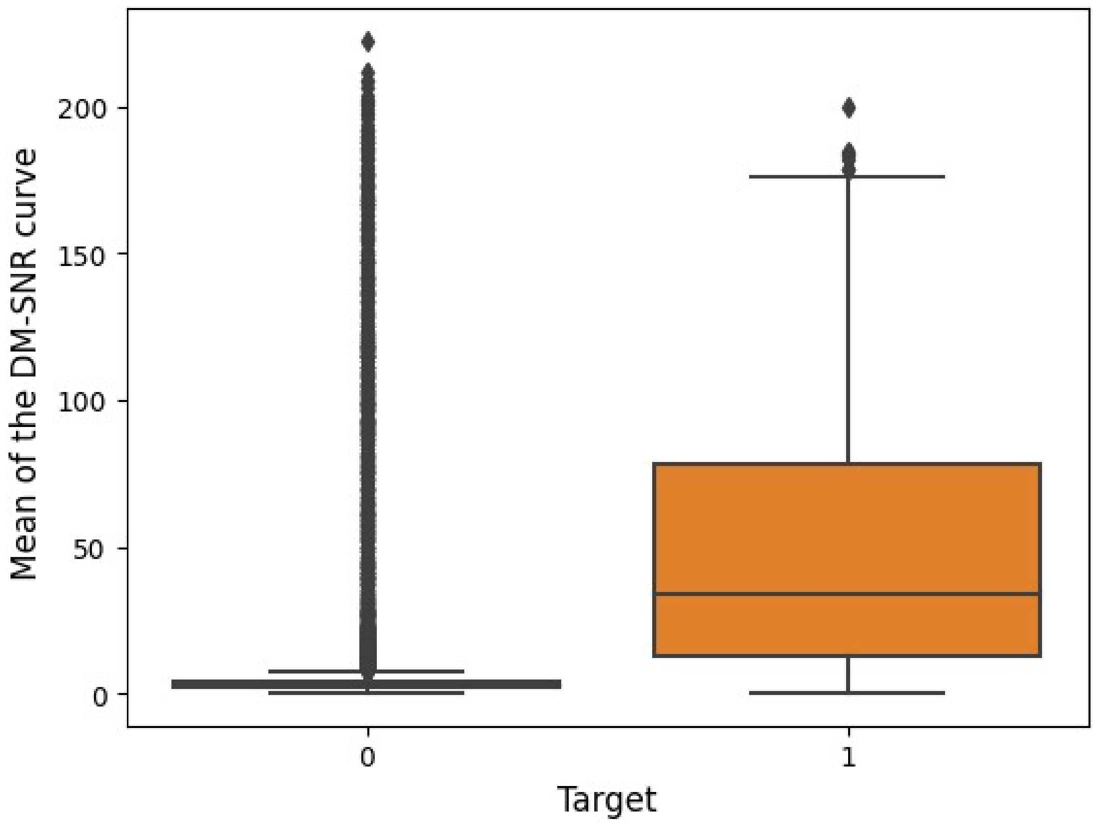

```
sns.boxplot(df, y = ' Excess kurtosis of the integrated profile' ,x =
'target_class')
plt.ylabel(' Excess kurtosis of the integrated profile', fontsize =
12)
plt.xlabel('Target', fontsize = 12)

Text(0.5, 0, 'Target')
```



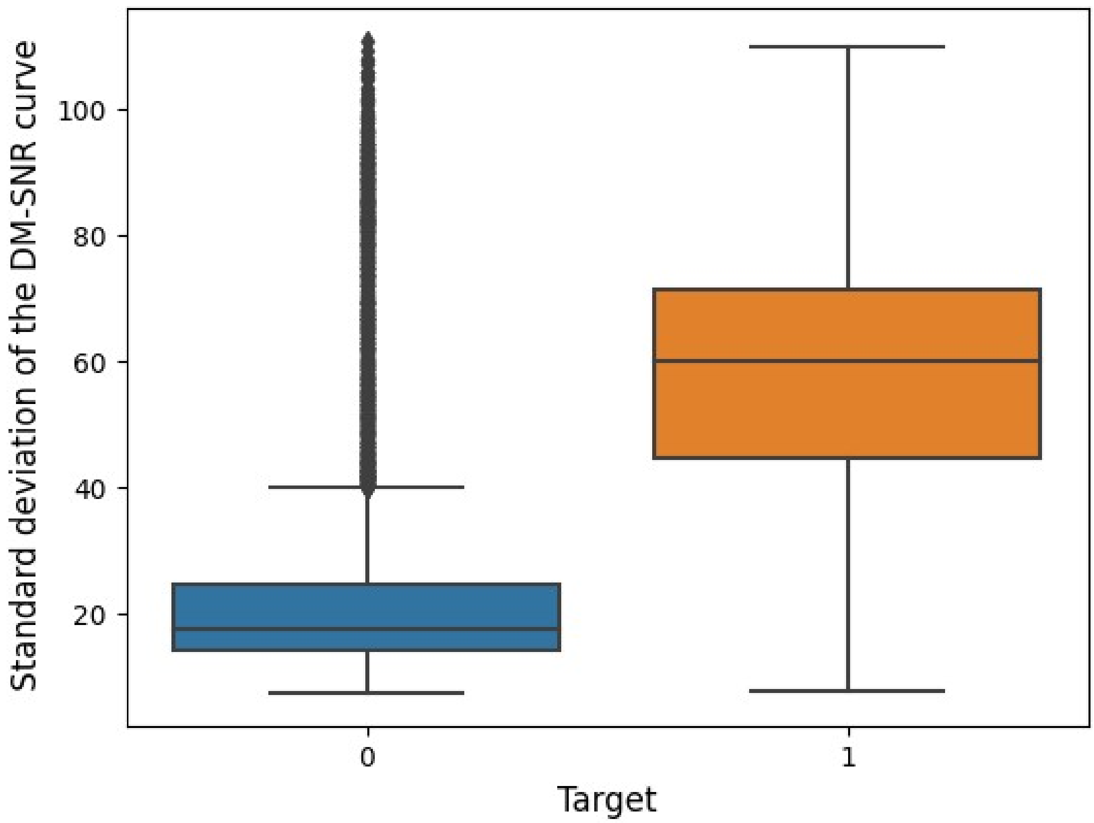

```
sns.boxplot(df, y = ' Skewness of the integrated profile' ,x =
'target_class')
plt.ylabel(' Skewness of the integrated profile', fontsize = 12)
plt.xlabel('Target', fontsize = 12)

Text(0.5, 0, 'Target')
```



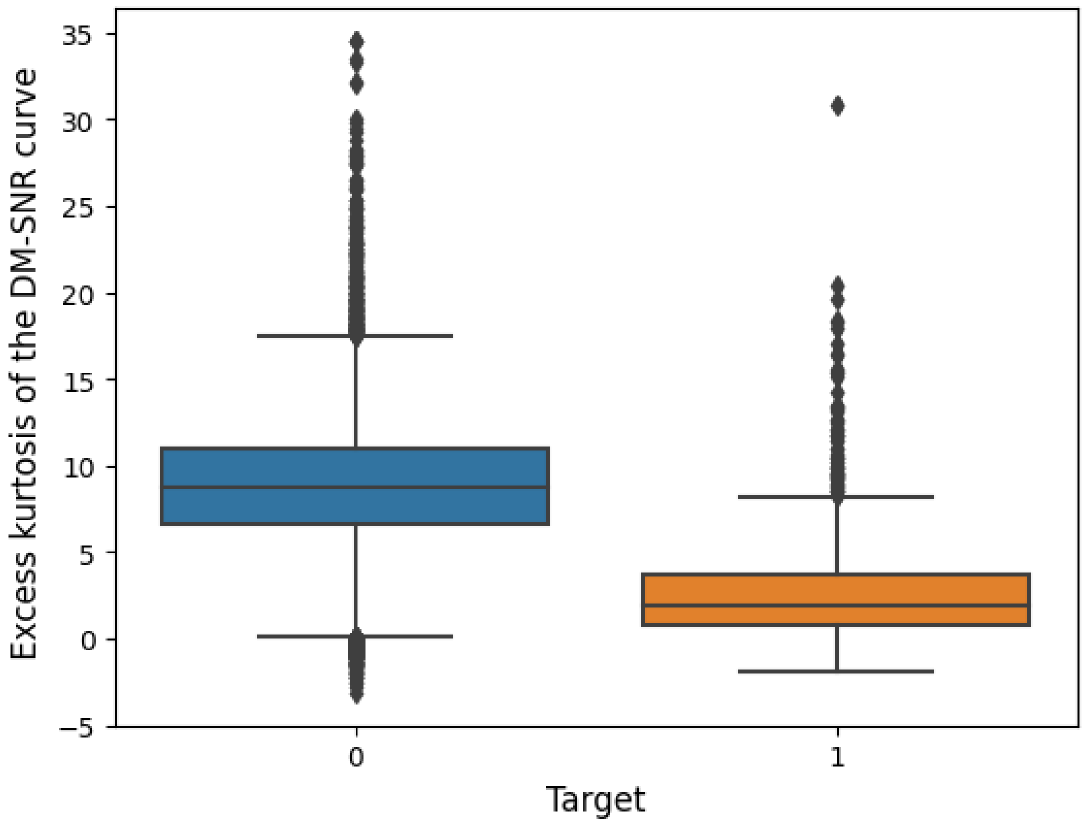

```
sns.boxplot(df, y = ' Mean of the DM-SNR curve' ,x = 'target_class')
plt.ylabel(' Mean of the DM-SNR curve', fontsize = 12)
plt.xlabel('Target', fontsize = 12)

Text(0.5, 0, 'Target')
```



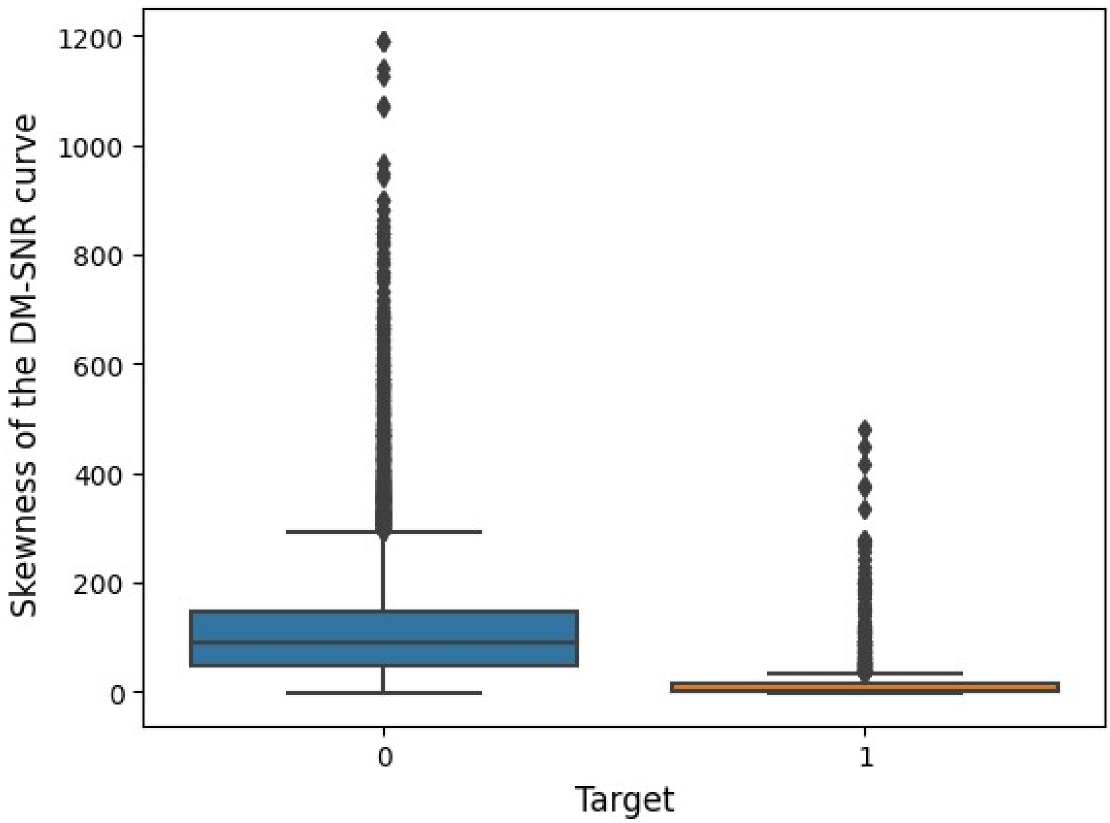

```
sns.boxplot(df, y = ' Standard deviation of the DM-SNR curve' ,x =
'target_class')
plt.ylabel(' Standard deviation of the DM-SNR curve', fontsize = 12)
plt.xlabel('Target', fontsize = 12)

Text(0.5, 0, 'Target')
```



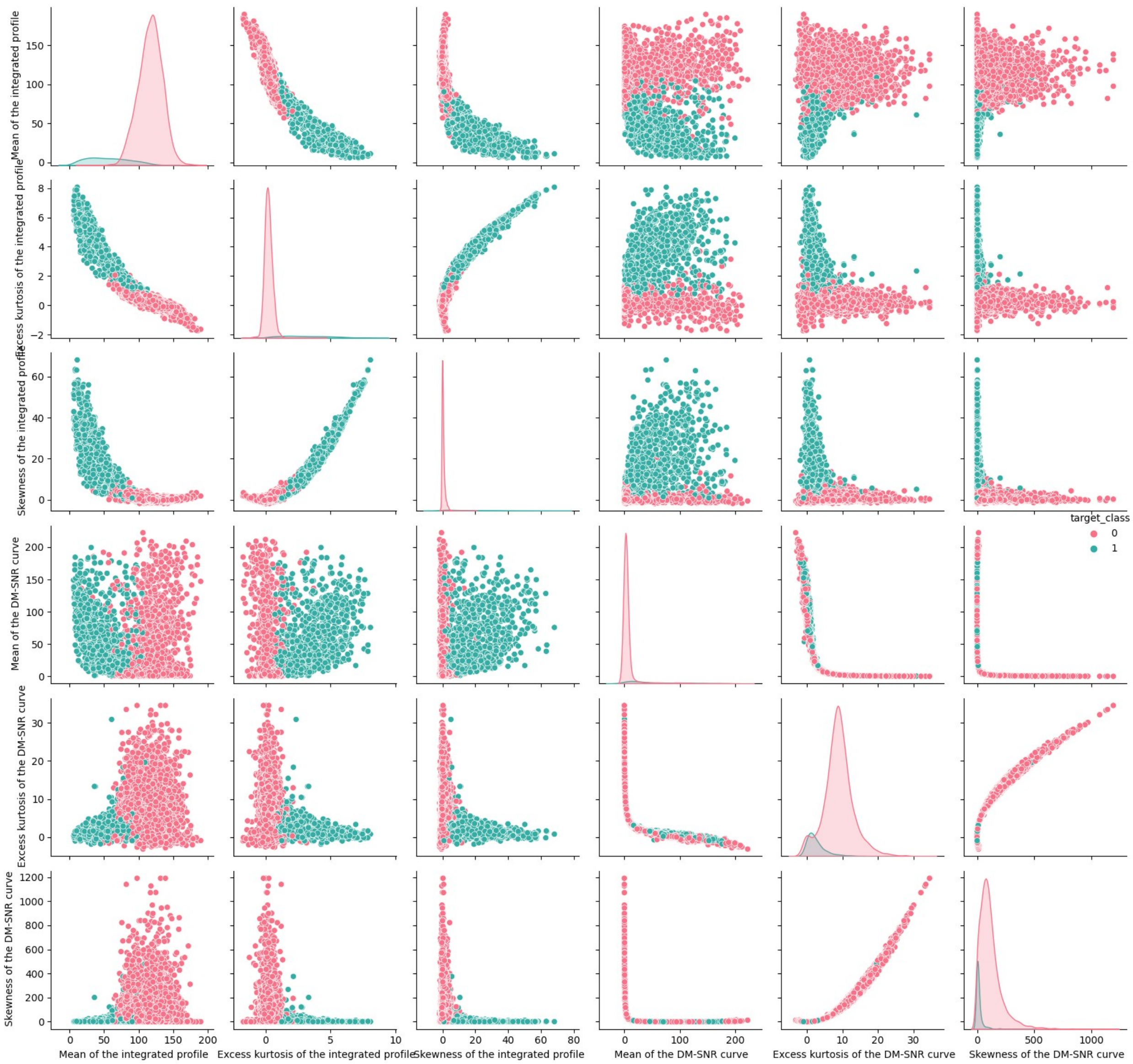

```
sns.boxplot(df, y = ' Excess kurtosis of the DM-SNR curve' ,x =
'target_class')
plt.ylabel(' Excess kurtosis of the DM-SNR curve', fontsize = 12)
plt.xlabel('Target', fontsize = 12)

Text(0.5, 0, 'Target')
```




```
sns.boxplot(df, y = ' Skewness of the DM-SNR curve' ,x =
'target_class')
plt.ylabel(' Skewness of the DM-SNR curve', fontsize = 12)
plt.xlabel('Target', fontsize = 12)

Text(0.5, 0, 'Target')
```

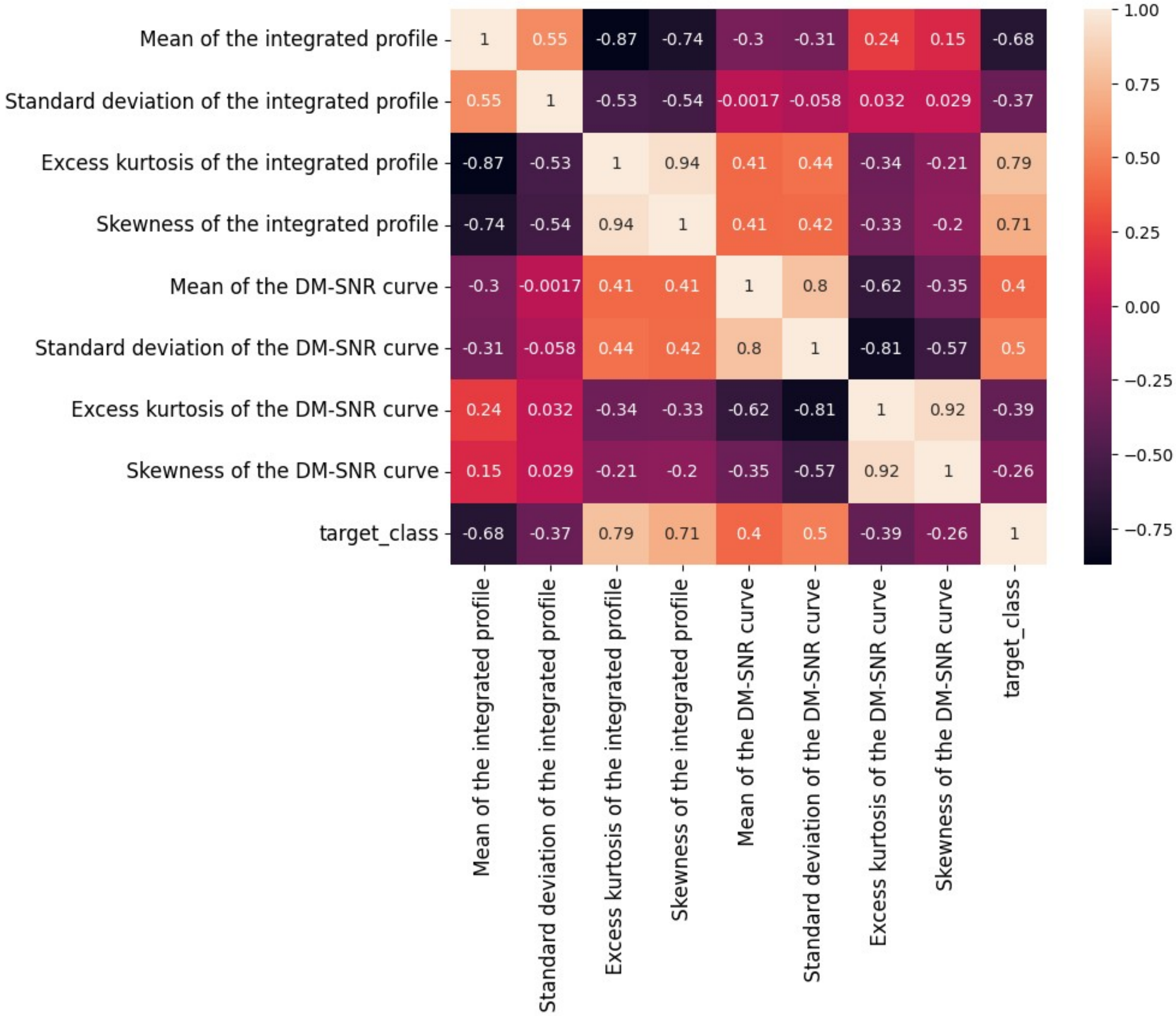



```
sns.pairplot(data=df,
             palette="husl",
             hue="target_class",
             vars=[" Mean of the integrated profile",
                   " Excess kurtosis of the integrated profile",
                   " Skewness of the integrated profile",
                   " Mean of the DM-SNR curve",
                   " Excess kurtosis of the DM-SNR curve",
                   " Skewness of the DM-SNR curve"])

# plt.suptitle("PairPlot of Data Without Std. Dev.
Fields",fontsize=18)

plt.tight_layout()
plt.show()
```




```
plt.figure(figsize = (8, 6))
sns.heatmap(df.corr(), annot = True)
plt.xticks(rotation = 90, fontsize = 12)
plt.yticks(rotation = 0, fontsize = 12)
```

```
(array([0.5, 1.5, 2.5, 3.5, 4.5, 5.5, 6.5, 7.5, 8.5]),
 [Text(0, 0.5, ' Mean of the integrated profile'),
  Text(0, 1.5, ' Standard deviation of the integrated profile'),
  Text(0, 2.5, ' Excess kurtosis of the integrated profile'),
  Text(0, 3.5, ' Skewness of the integrated profile'),
  Text(0, 4.5, ' Mean of the DM-SNR curve'),
  Text(0, 5.5, ' Standard deviation of the DM-SNR curve'),
  Text(0, 6.5, ' Excess kurtosis of the DM-SNR curve'),
  Text(0, 7.5, ' Skewness of the DM-SNR curve'),
  Text(0, 8.5, 'target_class')])
```



Observations

From the correlation heatmap it is clear that

1. Excess kurtosis of the integrated profile, Skewness of the integrated profile, and Standard deviation of the DM-SNR curve have high positive correlation (greater than 0.5) with the target class feature.
2. Apart from Mean of the DM-SNR curve, all reamining features have negative association with the target class feature.

Handling Missing Values

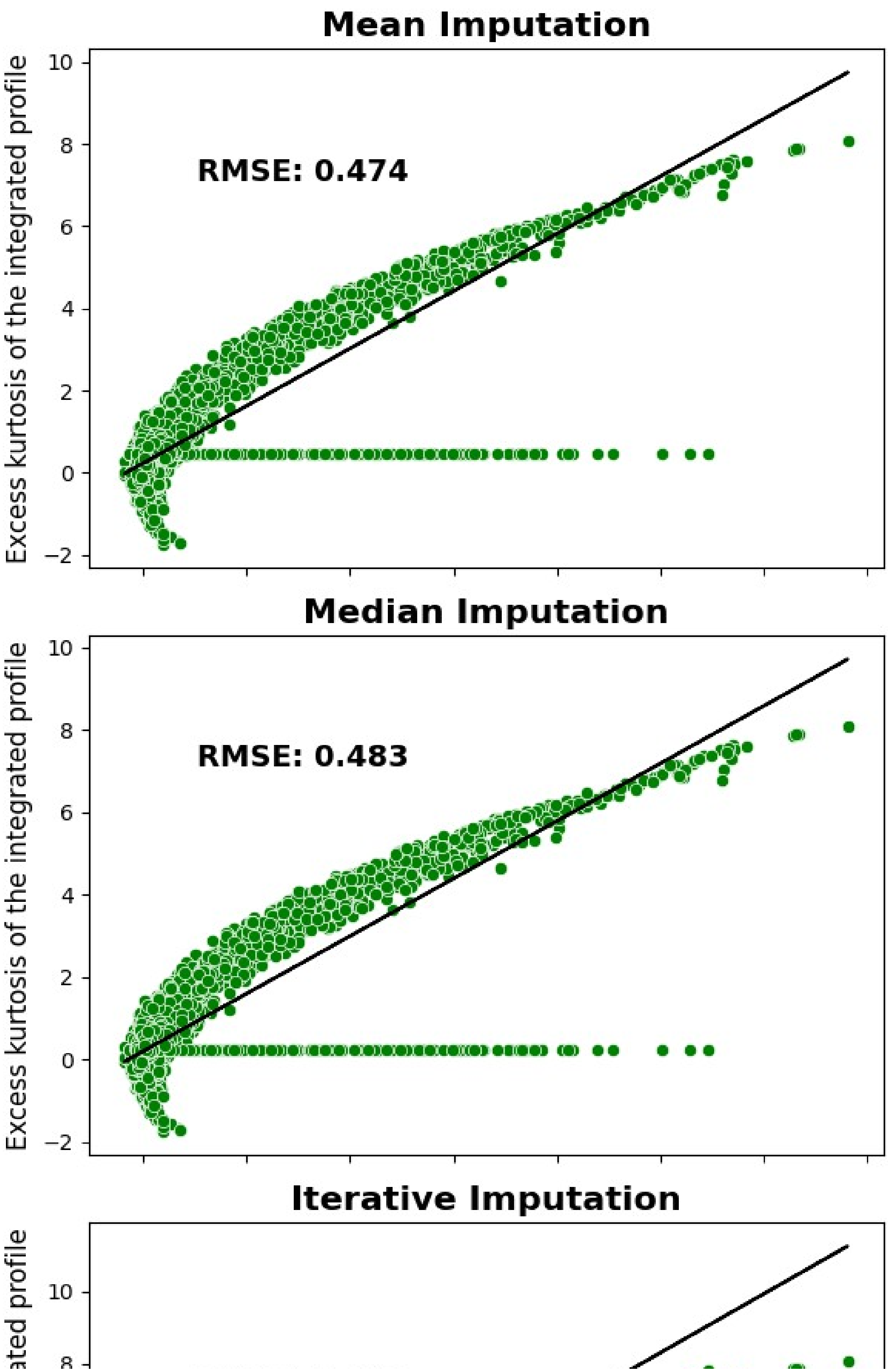
```
# Median Imputation
median_imputation = SimpleImputer(strategy = 'median')
median_imputed = median_imputation.fit_transform(df)
df_median_imputed = pd.DataFrame(median_imputed, columns = df.columns)
```

```
# Mean Imputation
mean_imputation = SimpleImputer(strategy = 'mean')
mean_imputed = mean_imputation.fit_transform(df)
df_mean_imputed = pd.DataFrame(mean_imputed, columns = df.columns)

# Iterative Imputation
iter_imputer = IterativeImputer(random_state=42)
iter_imputed = iter_imputer.fit_transform(df)
df_iter_imputed = pd.DataFrame(iter_imputed, columns = df.columns)
```

For verification of the imputations and plotting I have opted ' Excess kurtosis of the integrated profile' and ' Skewness of the integrated profile' because they have high positive association of 0.79, which can be seen in the plots.

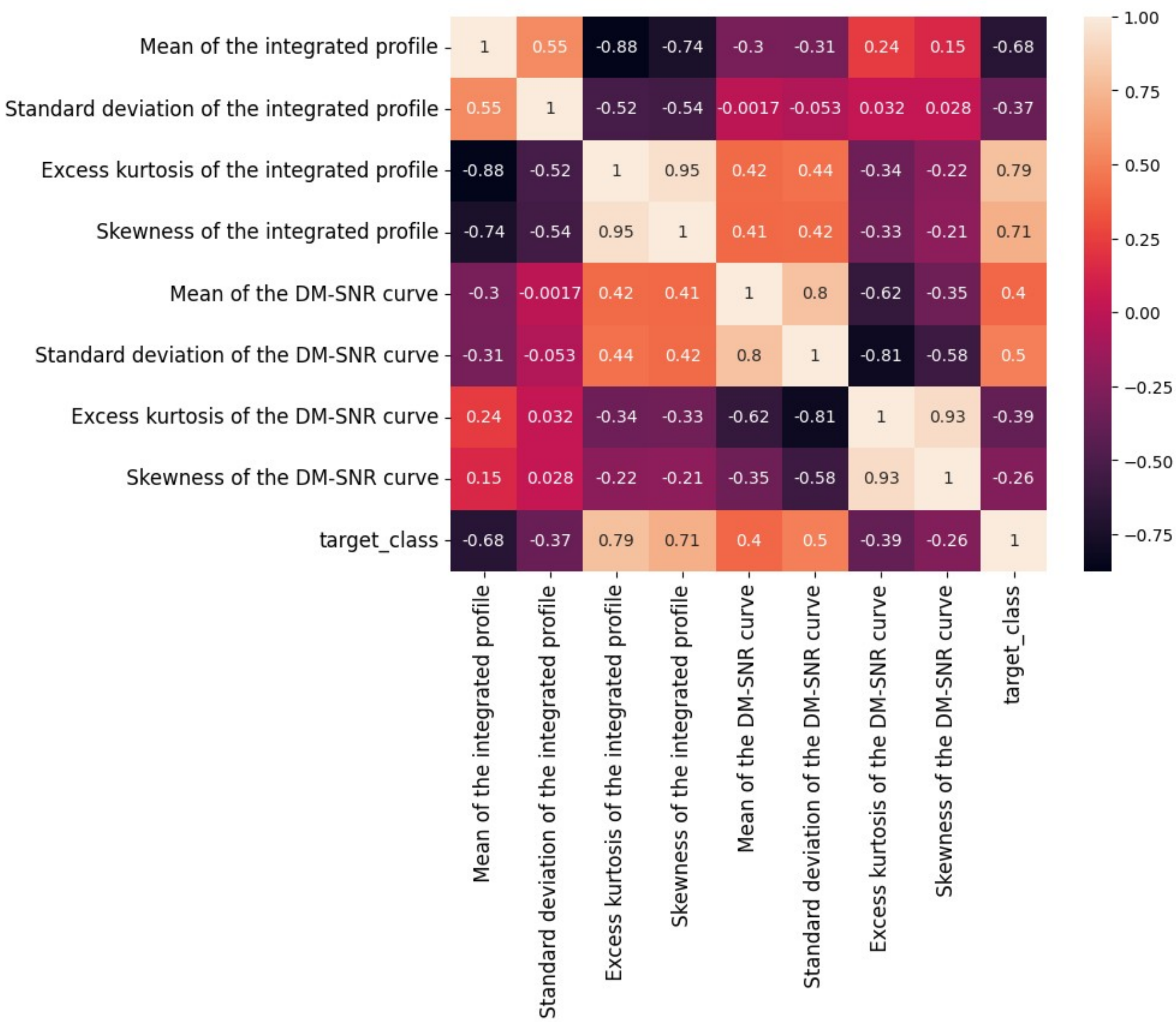
```
fig, axes = plt.subplots(nrows=3, ncols=1, sharex=True, figsize = (6,
12))
axes = np.reshape(axes, -1)

dfs = [df_mean_imputed, df_median_imputed, df_iter_imputed]
titles = ['Mean Imputation', 'Median Imputation', 'Iterative
Imputation']

for i, df in enumerate(dfs):
    # Plotting the data
    x = df[' Skewness of the integrated profile']
    y = df[' Excess kurtosis of the integrated profile']
    sns.scatterplot(x=x, y=y, ax=axes[i], color='green')

    # Fitting and plotting a linear regression line
    m, b = np.polyfit(x, y, 1)
    linreg = m * x + b
    axes[i].plot(x, linreg, color='black')

    # Setting the titles and including the RMSE values
    axes[i].set_title(titles[i], fontsize=16, fontweight='bold')
    rmse = round(mean_squared_error(y, linreg, squared=False), 3)
    text_x = min(x) + 0.1 * (max(x) - min(x))
    text_y = min(y) + 0.9 * (max(y) - min(y))
    axes[i].text(text_x, text_y, f'RMSE: {rmse}', fontsize=14,
fontweight='bold')

    # Set y-axis label
    axes[i].set_ylabel("Excess kurtosis of the integrated profile",
fontsize = 12)

# Set a common x-axis label
axes[-1].set_xlabel("Skewness of the integrated profile", fontsize =
12)
```

```
plt.tight_layout()
plt.show()
```


Mean Imputation
RMSE: 0.474
Excess kurtosis of the integrated profile
10
8
6
4
2
0
−2
Median Imputation
RMSE: 0.483
Excess kurtosis of the integrated profile
10
8
6
4
2
0
−2
Iterative Imputation
ated profile
10
8

```python
plt.figure(figsize = (8, 6))
sns.heatmap(df_iter_imputed.corr(), annot = True)
plt.xticks(rotation = 90, fontsize = 12)
plt.yticks(rotation = 0, fontsize = 12)
```

```
(array([0.5, 1.5, 2.5, 3.5, 4.5, 5.5, 6.5, 7.5, 8.5]),
 [Text(0, 0.5, ' Mean of the integrated profile'),
  Text(0, 1.5, ' Standard deviation of the integrated profile'),
  Text(0, 2.5, ' Excess kurtosis of the integrated profile'),
  Text(0, 3.5, ' Skewness of the integrated profile'),
  Text(0, 4.5, ' Mean of the DM-SNR curve'),
  Text(0, 5.5, ' Standard deviation of the DM-SNR curve'),
  Text(0, 6.5, ' Excess kurtosis of the DM-SNR curve'),
  Text(0, 7.5, ' Skewness of the DM-SNR curve'),
  Text(0, 8.5, 'target_class')])
```



Train Validation split

```
X = df_iter_imputed.drop('target_class', axis = 1)
y = df_iter_imputed['target_class']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 
0.2, random_state = 42)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

X_train = pd.DataFrame(X_train, columns = list(X.columns))
X_test = pd.DataFrame(X_test, columns = list(X.columns))

X_train.head()
```

```
    Mean of the integrated profile  \
0                        -0.014800   
1                         0.656612   
2                         0.519490   
3                        -0.135008   
4                         0.837226   

    Standard deviation of the integrated profile  \
0                                       0.008072   
1                                      -0.935945   
2                                       0.077371   
3                                      -0.117301   
4                                      -0.515738   

    Excess kurtosis of the integrated profile  \
0                                   -0.200569   
1                                   -0.434856   
2                                   -0.382747   
3                                   -0.093828   
4                                   -0.549088   

    Skewness of the integrated profile   Mean of the DM-SNR curve  \
0                            -0.267138                  -0.327459   
1                            -0.182541                  -0.269155   
2                            -0.261074                  -0.384134   
3                            -0.159627                  -0.362088   
4                            -0.214350                  -0.351977   

    Standard deviation of the DM-SNR curve  \
0                                -0.345614   
1                                -0.141328   
2                                -0.600240   
```

```
3                                -0.620212
4                                -0.410801
```

	Excess kurtosis of the DM-SNR curve	Skewness of the DM-SNR curve
0	-0.060276	-0.265333
1	-0.547480	-0.631169
2	1.028518	0.732324
3	0.469943	0.272002
4	0.208092	-0.103956

SVM with Default Parameters

```
svc = SVC()
svc.fit(X_train, y_train)
y_pred = svc.predict(X_test)
print(f'The accuracy of the model with default parameters is: 
{round(accuracy_score(y_test, y_pred), 4)}')

The accuracy of the model with default parameters is: 0.9816

# Checking for overfitting and underfitting
print(f'Training data score: {round(svc.score(X_train, y_train), 4)}')
print(f'Test data score: {round(svc.score(X_test, y_test), 4)}')

Training data score: 0.9796
Test data score: 0.9816
```

Since both the scores are comparable so there are no chance of overfitting or underfitting.

So, the models accuracy is 0.9816, let's very this score with null accuracy.

```
y_test.value_counts()

0.0    2279
1.0     227
Name: target_class, dtype: int64

y_test.size

2506
```

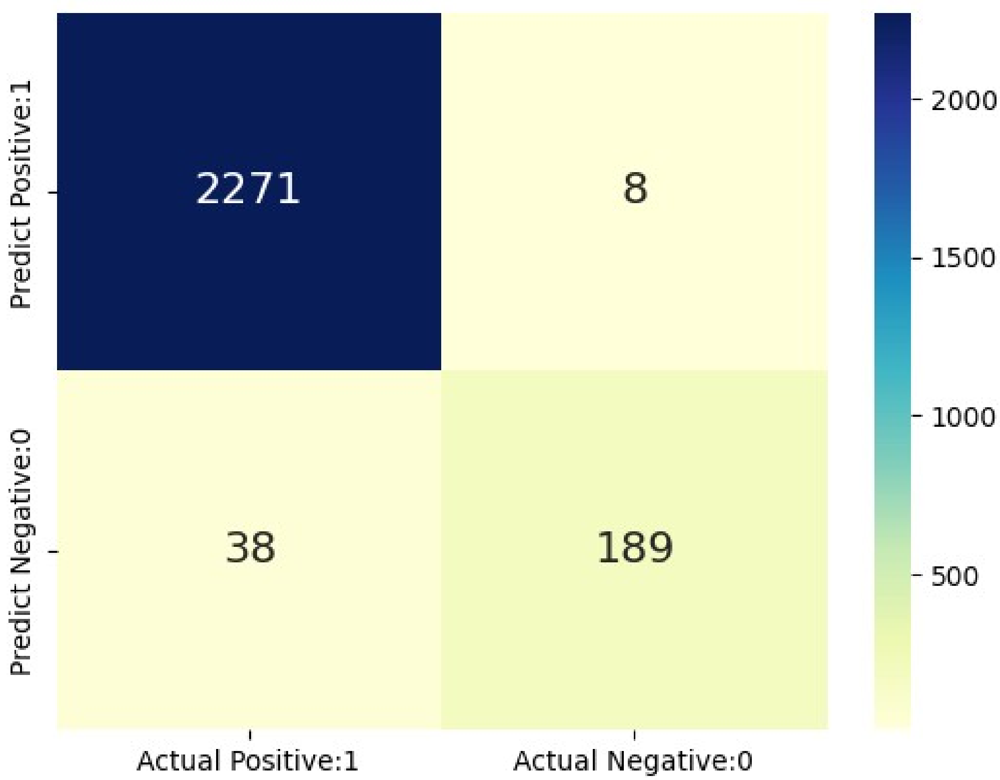
```python
# Since 0 is the most frequent class

null_accuracy = (y_test == 0).sum() / y_test.size
print(f'The null accuracy of the model is: {round(null_accuracy, 4)}')
```

```
The null accuracy of the model is: 0.9094
```

So, the accuracy achieved by the model is greater than that of null accuracy hence our model is doing well.

```python
cm = confusion_matrix(y_test, y_pred)
data_cm = pd.DataFrame(data=cm, columns=['Actual Positive:1', 'Actual Negative:0'],
                                 index=['Predict Positive:1', 'Predict Negative:0'])
sns.heatmap(data_cm, annot = True, fmt='d', cmap='YlGnBu', annot_kws = {'size': 16})
```

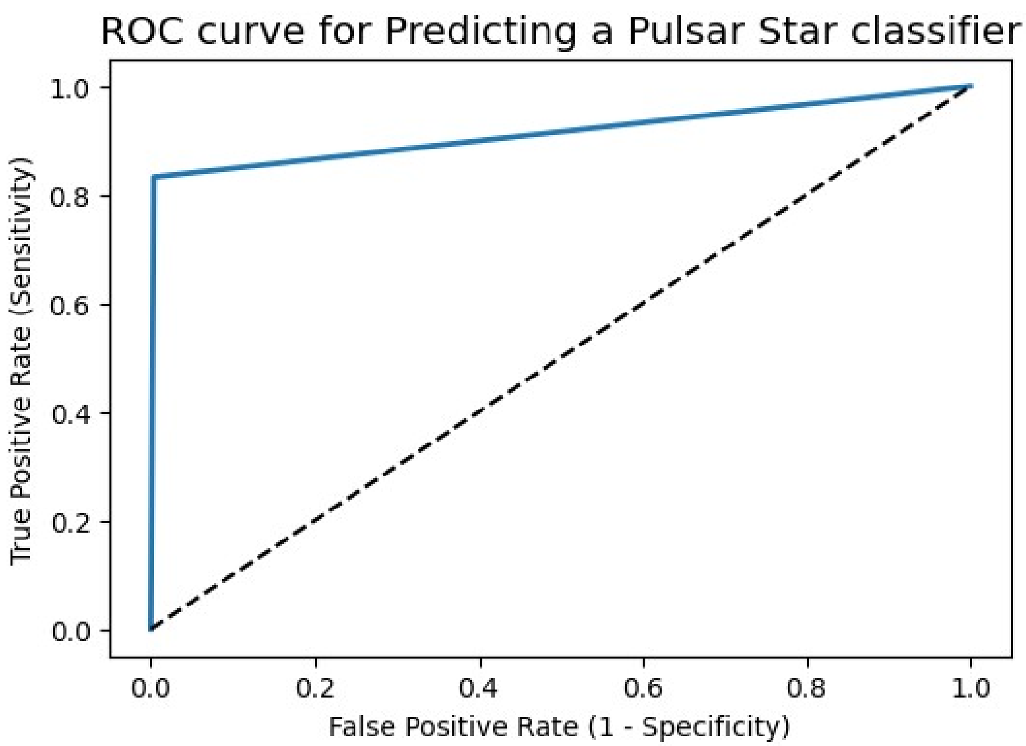
```
<Axes: >
```




```python
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.98      1.00      0.99      2279
```

```
         1.0       0.96      0.83      0.89       227

    accuracy                           0.98      2506
   macro avg       0.97      0.91      0.94      2506
weighted avg       0.98      0.98      0.98      2506


fpr, tpr, thresholds = roc_curve(y_test, y_pred)

plt.figure(figsize=(6,4))
plt.plot(fpr, tpr, linewidth=2)
plt.plot([0,1], [0,1], 'k--' )
plt.rcParams['font.size'] = 12
plt.title('ROC curve for Predicting a Pulsar Star classifier')
plt.xlabel('False Positive Rate (1 - Specificity)')
plt.ylabel('True Positive Rate (Sensitivity)')
plt.show()
```




```
print(f'The accuracy of the model is: {round(accuracy_score(y_test,
y_pred), 4)}')
print(f'The precision of the model is: {round(precision_score(y_test,
y_pred), 4)}')
print(f'The f1 score of the model is: {round(f1_score(y_test, y_pred),
4)}')
print(f'The ROC AUC score of the model is:
{round(roc_auc_score(y_test, y_pred), 4)}')
```

```
The accuracy of the model is: 0.9816
The precision of the model is: 0.9594
The f1 score of the model is: 0.8915
The ROC AUC score of the model is: 0.9145
```

Hyper parameter tuning

```
parameters = [
    {'C': [1, 10, 100, 1000], 'kernel': ['linear']},
    {'C': [1, 10, 100, 1000], 'kernel': ['rbf'], 'gamma': [0.1, 0.2,
0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]},
    {'C': [1, 10, 100, 1000], 'kernel': ['sigmoid'], 'gamma': [0.1,
0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]},
    {'C': [1, 10, 100, 1000], 'kernel': ['poly'], 'degree': [2, 3, 4,
5], 'gamma': [0.01, 0.02, 0.03, 0.04, 0.05]}
]


grid_search = GridSearchCV(estimator = svc, param_grid = parameters,
scoring = 'accuracy')
grid_search.fit(X_train, y_train)
```

```
GridSearchCV(estimator=SVC(),
             param_grid=[{'C': [1, 10, 100, 1000], 'kernel':
['linear']},
                         {'C': [1, 10, 100, 1000],
                          'gamma': [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7,
0.8,
                                    0.9],
                          'kernel': ['rbf']},
                         {'C': [1, 10, 100, 1000],
                          'gamma': [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7,
0.8,
                                    0.9],
                          'kernel': ['sigmoid']},
                         {'C': [1, 10, 100, 1000], 'degree': [2, 3, 4,
5],
                          'gamma': [0.01, 0.02, 0.03, 0.04, 0.05],
                          'kernel': ['poly']}],
             scoring='accuracy')
```

```
# best score achieved during the GridSearchCV
print('GridSearch CV best score : {:.4f}\n\
n'.format(grid_search.best_score_))


# print parameters that give the best results
print('Parameters that give the best results :','\n\n',
(grid_search.best_params_))
```

```
# print estimator that was chosen by the GridSearch
print('\n\nEstimator that was chosen by the search :','\n\n',
(grid_search.best_estimator_))
```

```
GridSearch CV best score : 0.9816


Parameters that give the best results :

 {'C': 10, 'kernel': 'linear'}


Estimator that was chosen by the search :

 SVC(C=10, kernel='linear')
```

```
# calculate GridSearch CV score on test set

print('GridSearch CV score on test set:
{0:0.4f}'.format(grid_search.score(X_test, y_test)))
```

```
GridSearch CV score on test set: 0.9832
```

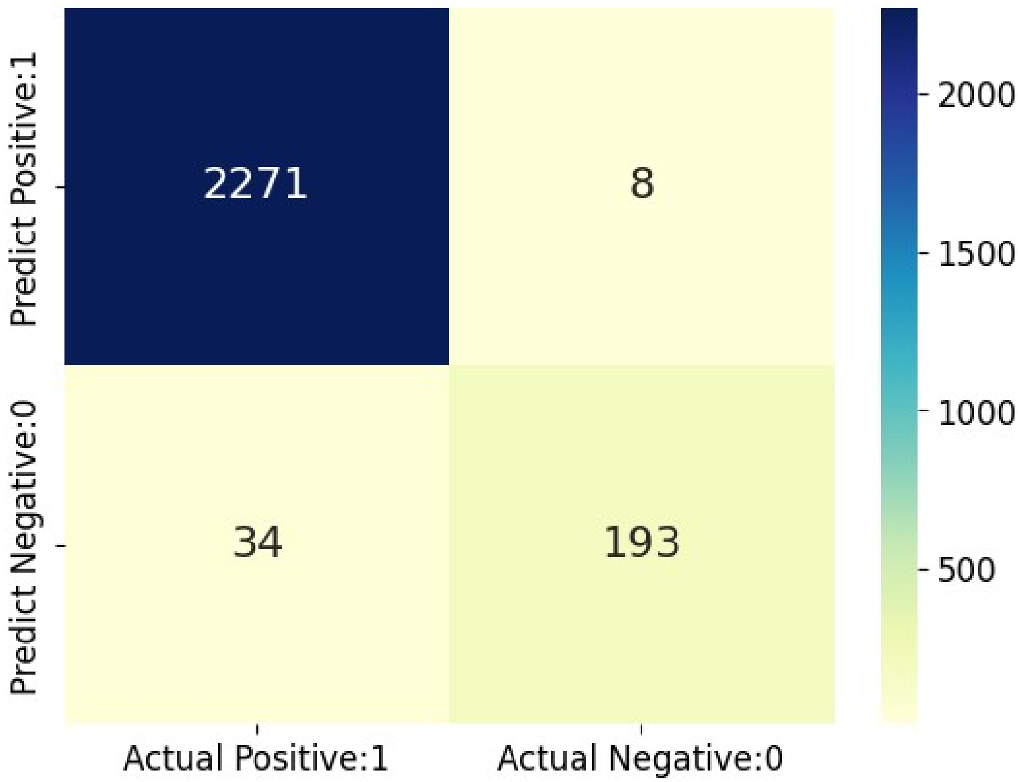
Making model with best estimators

```
svc_new = SVC(C = 10, kernel = 'linear')
svc_new.fit(X_train, y_train)
y_pred_new = svc_new.predict(X_test)
print(f'The accuracy of the model with default parameters is:
{round(accuracy_score(y_test, y_pred_new), 4)}')
```

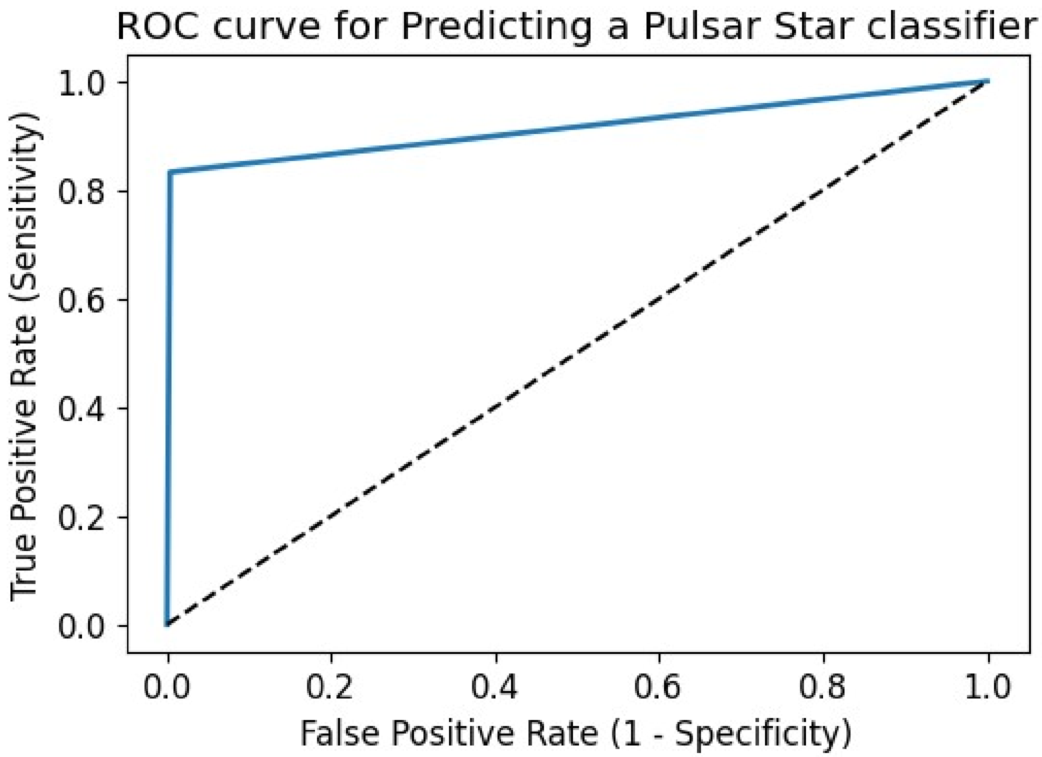
```
The accuracy of the model with default parameters is: 0.9832
```

```
cm = confusion_matrix(y_test, y_pred_new)
data_cm = pd.DataFrame(data=cm, columns=['Actual Positive:1', 'Actual
Negative:0'],
                                 index=['Predict Positive:1', 'Predict
Negative:0'])
sns.heatmap(data_cm, annot = True, fmt='d', cmap='YlGnBu', annot_kws =
{'size': 16})
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.98      1.00      0.99      2279
         1.0       0.96      0.83      0.89       227

    accuracy                           0.98      2506
   macro avg       0.97      0.91      0.94      2506
weighted avg       0.98      0.98      0.98      2506
```




```
print(f'The accuracy of the model is: {round(accuracy_score(y_test,
y_pred_new), 4)}')
print(f'The precision of the model is: {round(precision_score(y_test,
y_pred_new), 4)}')
print(f'The f1 score of the model is: {round(f1_score(y_test,
y_pred_new), 4)}')
print(f'The ROC AUC score of the model is:
{round(roc_auc_score(y_test, y_pred_new), 4)}')

The accuracy of the model is: 0.9832
The precision of the model is: 0.9602
The f1 score of the model is: 0.9019
The ROC AUC score of the model is: 0.9234

fpr, tpr, thresholds = roc_curve(y_test, y_pred)

plt.figure(figsize=(6,4))
plt.plot(fpr, tpr, linewidth=2)
plt.plot([0,1], [0,1], 'k--' )
plt.rcParams['font.size'] = 12
plt.title('ROC curve for Predicting a Pulsar Star classifier')
plt.xlabel('False Positive Rate (1 - Specificity)')
plt.ylabel('True Positive Rate (Sensitivity)')
plt.show()
```



Implementing the model to the test data

```
test_data = pd.read_excel(test_data_path)
test_data.head()
```

```
   Mean of the integrated profile  \
0                      116.906250
1                       75.585938
2                      103.273438
3                      101.078125
4                      113.226562

   Standard deviation of the integrated profile  \
0                                     48.920605
1                                     34.386254
2                                     46.996628
3                                     48.587487
4                                     48.608804
```

```
    Excess kurtosis of the integrated profile  \
0                                    0.186046
1                                    2.025498
2                                    0.504295
3                                    1.011427
4                                    0.291538

    Skewness of the integrated profile   Mean of the DM-SNR curve  \
0                            -0.129815                   3.037625
1                             8.652913                   3.765050
2                             0.821088                   2.244983
3                             1.151870                  81.887960
4                             0.292120                   6.291806

    Standard deviation of the DM-SNR curve  \
0                                17.737102
1                                21.897049
2                                15.622566
3                                81.464136
4                                26.585056

    Excess kurtosis of the DM-SNR curve   Skewness of the DM-SNR curve
\
0                              8.122621                      78.813405

1                              7.048189                      55.878791

2                              9.330498                     105.134941

3                              0.485105                      -1.117904

4                              4.540138                      21.708268


   target_class
0           NaN
1           NaN
2           NaN
3           NaN
4           NaN

test_data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5370 entries, 0 to 5369
Data columns (total 9 columns):
 #   Column                                         Non-Null Count
Dtype
---  ------                                         --------------
-----
 0    Mean of the integrated profile                5370 non-null
```

```
float64
 1    Standard deviation of the integrated profile  5370 non-null
float64
 2    Excess kurtosis of the integrated profile     4603 non-null
float64
 3    Skewness of the integrated profile            5370 non-null
float64
 4    Mean of the DM-SNR curve                      5370 non-null
float64
 5    Standard deviation of the DM-SNR curve        4846 non-null
float64
 6    Excess kurtosis of the DM-SNR curve           5370 non-null
float64
 7    Skewness of the DM-SNR curve                  5126 non-null
float64
 8   target_class                                   0 non-null
float64
dtypes: float64(9)
memory usage: 377.7 KB
```

```
for items in list(test_data.columns):
  s = test_data[items].isna().sum()
  print(f'The number of missing values in {items} are {s}')
```

```
The number of missing values in  Mean of the integrated profile are 0
The number of missing values in  Standard deviation of the integrated
profile are 0
The number of missing values in  Excess kurtosis of the integrated
profile are 767
The number of missing values in  Skewness of the integrated profile
are 0
The number of missing values in  Mean of the DM-SNR curve are 0
The number of missing values in  Standard deviation of the DM-SNR
curve are 524
The number of missing values in  Excess kurtosis of the DM-SNR curve
are 0
The number of missing values in  Skewness of the DM-SNR curve are 244
The number of missing values in target_class are 5370
```

```
X_test_data = test_data.drop('target_class', axis = 1)

iter_imp = IterativeImputer(random_state = 42)
imputed = iter_imp.fit_transform(X_test_data)
df_iterative_imputed = pd.DataFrame(imputed, columns =
list(X.columns))

scaled = scaler.fit_transform(df_iterative_imputed)
df = pd.DataFrame(scaled, columns =
list(df_iterative_imputed.columns))
y_pred_test_data = svc_new.predict(df)
```

```
print(np.unique(y_pred_test_data))
```

```
[0. 1.]
```

```
print(f'The accuracy score is {svc.score(df, y_pred_test_data)}')
```

```
The accuracy score is 0.9945996275605214
```

```
df_iterative_imputed['target_class'] = pd.DataFrame(y_pred_test_data)
```

```
df_iterative_imputed.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5370 entries, 0 to 5369
Data columns (total 9 columns):
 #   Column                                         Non-Null Count
Dtype
---  ------                                         --------------
-----
 0    Mean of the integrated profile                5370 non-null
float64
 1    Standard deviation of the integrated profile  5370 non-null
float64
 2    Excess kurtosis of the integrated profile     5370 non-null
float64
 3    Skewness of the integrated profile            5370 non-null
float64
 4    Mean of the DM-SNR curve                      5370 non-null
float64
 5    Standard deviation of the DM-SNR curve        5370 non-null
float64
 6    Excess kurtosis of the DM-SNR curve           5370 non-null
float64
 7    Skewness of the DM-SNR curve                  5370 non-null
float64
 8   target_class                                   5370 non-null
float64
dtypes: float64(9)
memory usage: 377.7 KB
```

```
df_iterative_imputed['target_class'].value_counts()
```

```
0.0    4944
1.0     426
Name: target_class, dtype: int64
```

```
null_accur = (df_iterative_imputed['target_class'] == 0.0).sum() /
df_iterative_imputed['target_class'].size
print(f'The null accuracy obtained by the model = {null_accur}')
```

```
The null accuracy obtained by the model = 0.9206703910614525
```