Assignment 6
A mathematical essay on Support Vector Machine

Anik Bhowmick
Inter Disciplinary Dual Degree Data-Science
Indian Institute of Technology Madras
ae20b102@smail.iitm.ac.in

***Abstract*—This assignment discusses the application of the Support Vector Machine on the Pulsar Data Set. HTRU2 actually created this data. The data comprises various kinds of star features, such as the Mean of the integrated profile, Standard deviation of the integrated profile, Excess kurtosis of the integrated profile, etc. Based on it, the task is to predict whether a star is a pulsar star or not. Pulsars are a rare type of Neutron star. The electromagnetic spectrum produced by them is detectable from the earth. Currently, various machine learning techniques are employed to automate the detection of this kind of stars. These stars are very important for current scientific research, study of interstellar medium and many more. The present dataset is purely numerical type, with a target column having binary entries. However, it contains lots of missing entries, so various null value imputation techniques will be discussed in this paper, in addition to a detailed mathematical analysis of the SVM classifier.**

***Index Terms*—Visualization, SVM classifier, Correlation coefficient, Confusion Matrix, Accuracy, Precision, Recall, F_1 score, ROC.**

I. Introduction

- This assignment is a binary classifier about whether a neutron star is a pulsar star or not using the Support Vector Machine classifier. The dataset used for this purpose is the Pulsar Star Data Set. The main task is to unfold the underlying features in the data that decide the nature of a star.
- A SVM is a supervised machine learning algorithm mainly used for classification tasks where the goal is to predict whether an instance belongs to a given class or not. This algorithm can be used for regression also. They work by constructing a hyperplane that best separates the dataset into two classes. A hyperplane is a higher-dimensional version of a 2-dimensional plane. The dimension of the hyperplane is 1 less than the feature dimension of the data.
- The main task in this assignment is to build a binary classifier model to classify stars by looking into their various attributes; for this purpose, the SVM algorithm will be used.
- This paper will demonstrate the data analysis technique relevant to the SVM classifier with the help of visual plots and mathematical equations. This paper covers various data handling techniques, the model's fitting and validation, the algorithm's working principle, and the mathematics behind various evaluation metrics.

II. Support Vector Machine

It is a maximum margin classifier whose margins are defined by specific data points called Support vectors. That's why it is called a support vector machine. Support vector machine is a binary classifier by default but can be extended to multiclass classification. But multiclass classification is beyond the scope of this paper as the present problem is a binary classification problem. SVM generates a hyperplane to separate the data, so in some sense, it is a linear classifier. But with the help of kernels, it can generate non-linear hyper-surface. We will discuss kernels briefly in the kernel subsection. Some of the important terminologies are given below.

1. **Support Vectors:**
   - **Definition:** Support vectors are the data points that are closest to the decision boundary (or hyperplane) of a Support Vector Machine (SVM). They are pivots in defining the decision boundary.
   - **Importance:** These points are the most challenging instances to classify and are crucial for determining the optimal hyperplane that maximizes the margin.
2. **Margin:**
   - **Definition:** The margin is the distance between the decision boundary (hyperplane) and the nearest data point from either class. In SVM, the goal is to find the hyperplane that maximizes this margin.
   - **Importance:** A larger margin indicates a more robust and generalizable model. SVM aims to find the hyperplane that separates the classes with the maximum margin, resulting in better performance on unseen data.
3. **Hard Margin**
   - **Definition:** In a hard-margin SVM, the algorithm enforces a strict margin, requiring the data points to be perfectly separated without any misclassifications. This approach is suitable for linearly separable and noise-free data.
4. **Soft Margin**
   - **Definition:** In a soft-margin SVM, the algorithm allows for some misclassifications and overlapping of classes to

achieve a balance between maximizing the margin and tolerating errors. It is more flexible and robust when dealing with noisy or overlapping data.

A regularization parameter called C. is used to make an SVM classifier as hard or soft margin.

A. Assumptions

Some of the assumptions made by SVM are:

- It assumes data points are linearly separable while using linear kernels
- For datapoints that are not linearly separable it uses some special functions called kernels.
- SVM expects data to be properly scaled. But it can work with unscaled data too.

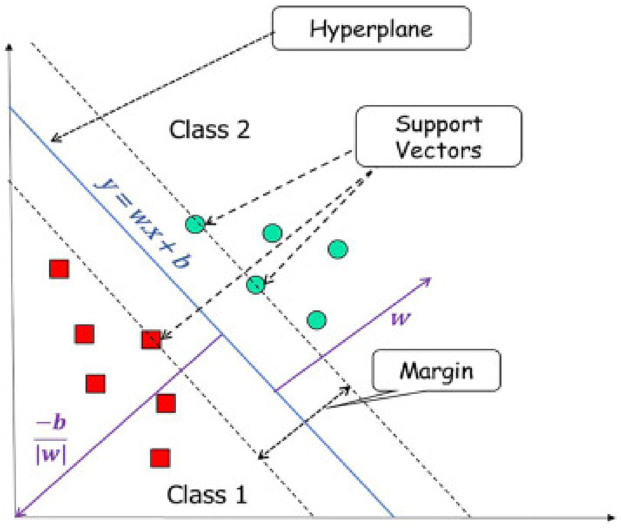


Fig. 1. SVM with margin

B. Hard and soft margin

A **Hyperplane** is a subspace of one dimension less than of its ambient space. A hyperplane in p dimension can be given as:

$$f(\mathbf{X}) = w_1x_1 + w_2x_2... + w_px_p + b = \mathbf{w.x_i} + b = 0$$

x_i is the i-th point. Consider a dataset of n training examples with binary labels $y_i \in \{-1, 1\}$ for two classes. The hyperplane should be of the following kind:

$$f(\mathbf{X}) > 0 \text{ if } y_i = +1$$

$$f(\mathbf{X}) < 0 \text{ if } y_i = -1$$

But there are infinitely many hyperplanes which can satisfy the property given above. In order to choose the optimal hyperplane, the concept margin comes into play. The signed distance of any point from the hyperplane can be given as:

$$d = \frac{\mathbf{w.x_i} + b}{||\mathbf{w}||}$$

For a given hyperplane, we consider that point to be its support vector for which the absolute value of the signed distance given above is the minimum. So:

$$\min_{i \in n} |d|$$

Or, we can find the minimum from the geometric margin given as:

$$M = \min_{i \in n} \frac{y_i(\mathbf{w.x_i} + b)}{||\mathbf{w}||}$$

But to get the optimal hyperplane from different hyperplanes, we look for the hyperplane having the maximum margin. So

$$\max_k M_k$$

The complete optimization problem can be given as

$$\max_{w,b} \frac{(\mathbf{w.x} + b)}{||\mathbf{w}||}$$

$$\text{subject to } \frac{y_i(\mathbf{w.x_i} + b)}{||\mathbf{w}||} \geq \frac{y_i(\mathbf{w.x} + b)}{||\mathbf{w}||}$$

Because for a given hyperplane, its support vectors are fixed by suitable choice of $\mathbf{w}$ and b, we can make $\mathbf{w.x} + b = 1$, in this case, the above minimization will become:

$$\max_{w,b} \frac{1}{||\mathbf{w}||}$$

$$\text{subject to } y_i(\mathbf{w.x_i} + b) \geq 1$$

We need one more modification to convert the above problem into convex optimization. It is found that this modification does not cause the optimal solution to change.

$$\min_{w,b} \frac{1}{2}||\mathbf{w}||^2$$

$$\text{subject to } y_i(\mathbf{w.x_i} + b) \geq 1$$

The optimization formulated above belongs to the hard margin classifier, where the classifier tries to construct complex decision boundaries to classify the data into the correct class. But in the real-world scenario, most of the time, the data is not linearly separable. In such cases, the notion of a soft margin comes into play. It is called soft because the separability of some of the training observations can be violated. It allows some data points to reside inside the margin or even on the wrong side of the hyperplane. This flexibility is crucial when dealing with real-world datasets that might contain noise or overlapping points. By incorporating a parameter called C, the soft margin SVM strikes a balance between maximizing the margin width and accommodating certain observations within or beyond the margin.

We introduce a slack variable ϵ_i. If a data point x_i falls outside the margin, $\epsilon_i = 0$, whereas $\epsilon_i > 0$ for support vectors. The optimization problem is now:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}|||\mathbf{w}|||^2 + C\sum_{i=1}^{n} \epsilon_i \\ \text{subject to} \quad & y_i(\mathbf{w.x_i} + b) \geq 1 - \epsilon_i \\ & \epsilon_i \geq 0 \quad \text{for } i = 1, 2, \ldots, n \end{aligned}$$

The hyperparameter C plays a crucial role in determining how strictly the SVM enforces the margin violations. When C is set to a very large value ($C \to \infty$), the SVM penalizes misclassifications heavily. As a result, the algorithm strives to create narrow margins that are rarely violated, making the classifier highly tailored to the training data. This approach might lead to a low-bias, high-variance model.

Conversely, when C is small, the SVM allows for more margin violations and misclassifications. In this scenario, wider margins are accepted, allowing the model to generalize more and be less influenced by individual data points. A small C value results in a high-bias, low-variance model.

In essence, C serves as a tuning parameter, striking a balance between bias and variance. A large C value leans towards overfitting by creating a complex decision boundary tailored to the training data, while a small C value encourages a simpler, more generalized decision boundary. The choice of C is often determined through techniques like cross-validation, ensuring the SVM finds an optimal trade-off between fitting the training data well and generalizing it to new, unseen data.

C. Solution of the optimizaton

The above optimization problem can be solved either by Lagrange primal or dual approach. The solution of the dual form is much easier than the primal form. The Dual form converts the above problem to a maximization problem in terms of Lagrange multipliers. The Lagrange form of the hard margin SVM is given as:

$$\mathcal{L} = \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{i=n} \alpha_i \left[y_i(\mathbf{w}.\mathbf{x_i} + b) - 1 \right]$$

Without going into the detailed derivation, the dual form can be given as:

$$W(\alpha) = \sum_{i=1}^{i=n} \alpha_i - \frac{1}{2} \sum_{i=1}^{i=n} \sum_{j=1}^{j=n} \alpha_i \alpha_j y_i y_j \mathbf{x_i}.\mathbf{x_j}$$

Now the optimization function is:

$$\max_{\alpha} \ W$$
$$\text{subject to } \alpha_i \geq 0$$
$$\sum_{i=1}^{i=n} \alpha_i y_i = 0$$

With a soft margin, the above problem converts to:

$$\max_{\alpha} \ W$$
$$\text{subject to } C \geq \alpha_i \geq 0$$
$$\sum_{i=1}^{i=n} \alpha_i y_i = 0$$

The decision function after solving we get as given below:

$$f(x) = \beta_0 + \sum_{i=1}^{i=n} \alpha_i < x, x_i >$$

Where $< x, x_i >$ is dot product and $\alpha_i \neq 0$ for support vectors only. Since the decision boundary is constructed only by support vectors. The SVM is quite insensitive to the outliers.

D. Kernels

The solution given above can be written for some kernels as:

$$f(x) = \beta_0 + \sum_{i=1}^{i=n} \alpha_i K(x, x_i)$$

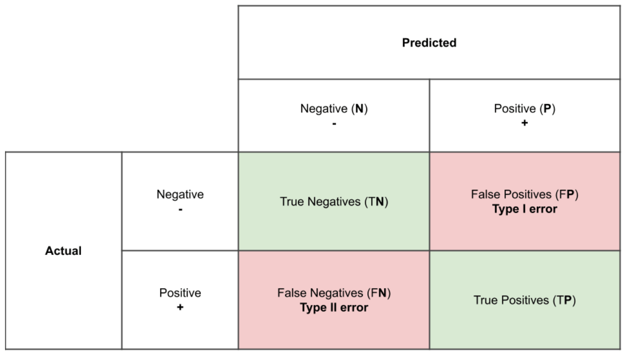
The solution in the previous equation is an example of a linear kernel. When the data points are not linearly separable, we need data representation in higher dimensional space. Kernels enable us to do so. The main advantage of using a kernel is that it makes the data separable with the help of generalized hyperspace. Some of the kernels are given below:

- **Linear kernel:** $K(xi, xj) = x_i^T xj$
- **Polynomial kernel:** $K(xi, xj) = (1 + x_i^T xj)^d$
- **RBF kernel:** $K(xi, xj) = e^{(-\gamma ||x_i - x_j||^2)}$

E. Evaluation Metrics

Evaluation metrics are useful for the assessment of a model after its training.

- **Confusion Matrix** A confusion matrix is the table often used to describe the performance of a classification model on a set of test data for which the true values are known. A confusion matrix looks exactly as given below.



		Predicted	
		Negative (**N**) -	Positive (**P**) +
Actual	Negative -	True Negatives (T**N**)	False Positives (F**P**) **Type I error**
	Positive +	False Negatives (F**N**) **Type II error**	True Positives (T**P**)

Fig. 2. Confusion matrix for multiclass classification

TP: True positive is how many positives are true as predicted by the model.
FP: False positive is how many are predicted to be positive but not positive.
FN: False negative is how many are falsely predicted to be negative.
TN: True negative is how many are predicted to be negative, which are actually negative.
Only TP and TN are the correct predictions made by the model. The rest are wrong predictions. A good model should have FP and FN as small as possible.

- **Accuracy** We define accuracy as the fraction of correct prediction out of the total prediction given by the model.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

A good model should have high accuracy. But that is not a necessary condition. Even a highly accurate model can give more wrong predictions if trained on a highly imbalanced dataset (i.e., the number of one class instances is much larger than the other). So, accuracy alone can not help us decide whether a model is good or not.

- **Precision**: This is defined as how many are actually positive out of total positive prediction.

$$\text{Precision} = \frac{TP}{TP + FP}$$

- **Recall**: or true positive rate is defined as the number of predicted positive out of actual positive. It is also known as Sensitivity.

$$\text{Recall (TPR)} = \frac{TP}{TP + FN}$$

- ***F_1* score** : Is the harmonic mean of precision and recall.

$$\boldsymbol{F_1} \textbf{ score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

Again, for F_1 score for each class, it can be defined.

III. The Data

Brief information about the data:

- Total data points are 12528 in the train and 5370 in the test set. Columns are 'Mean of the integrated profile', 'Standard deviation of the integrated profile', 'Excess kurtosis of the integrated profile', 'Excess kurtosis of the integrated profile', 'Excess kurtosis 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'.

Variable	Type
Mean of the integrated profile	Continuous
Standard deviation of the integrated profile	Continuous
Excess kurtosis of the integrated profile	Continuous
Skewness of the integrated profile	Continuous
Mean of the DM-SNR curve	Continuous
Standard deviation of the DM-SNR curve	Continuous
Excess kurtosis of the DM-SNR curve	Continuous
Skewness of the DM-SNR curve	Continuous
Target class	Class 0, 1

TABLE I
VARIABLE DEFINITIONS

IV. The Problem

As mentioned, our goal in this assignment is to predict whether a neutron star is pulsar or not.

A. *Cleaning and preparing the data*

This data is completely numerical, with some missing entries.

1) Missing Value imputation: The general approach for imputing missing values of continuous variables is either mean or median. Mean is a good choice when the distribution is close to normal. However, the median is preferred for replacing the missing values for a skewed dataset. Below are the three distribution plots for three features containing missing values.

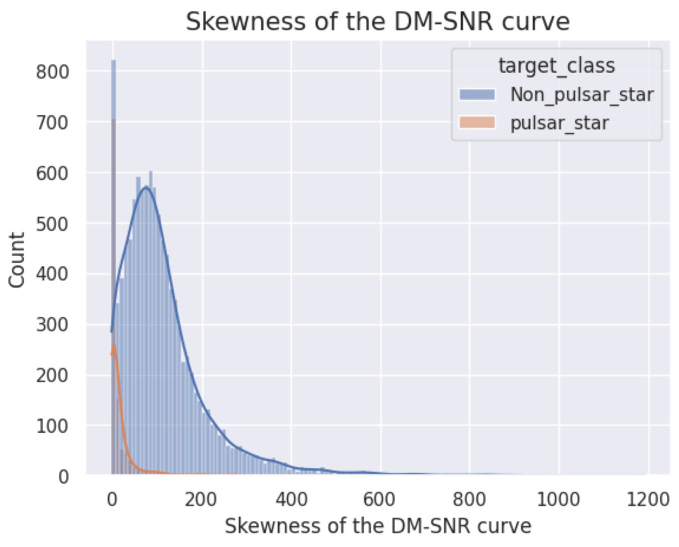


Fig. 3. Skewness of DM-SNR curve

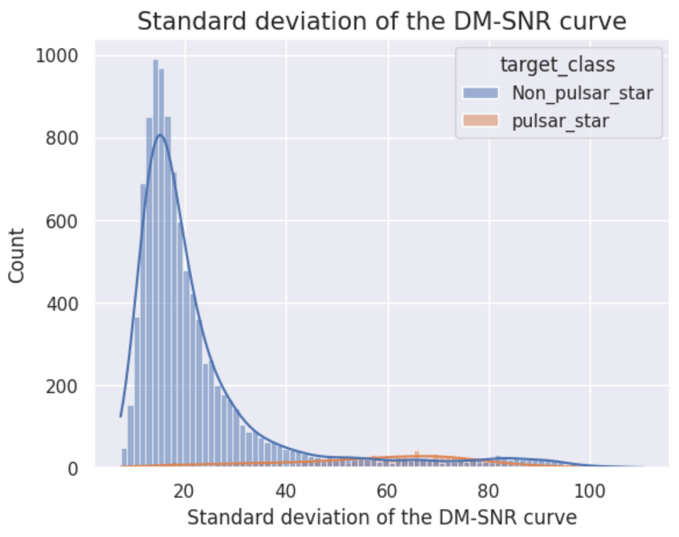


Fig. 4. Standard deviation of DM-SNR curve

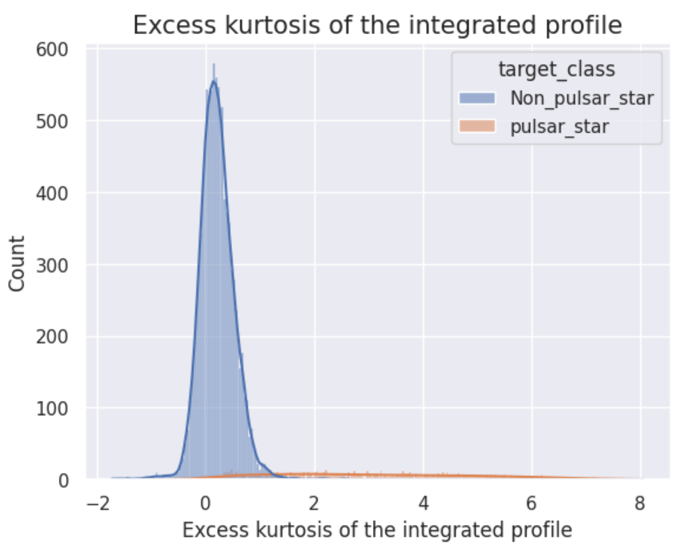


Fig. 5. Excess kurtosis of integrated profile

These plots are skewed, so median imputation is the best for each. But we used a different imputation technique, called K nearest neighbour imputation. KNN uses neighbouring data points to decide what value to put at the missing place. This is quite useful for numerical data. K-Nearest Neighbors (KNN) imputation is a method for filling in missing values in a dataset. It works by identifying the 'k' nearest neighbours to each missing value using a similarity metric. The missing value is then calculated based on the values of these neighbours, often using an average or weighted average approach. By leveraging the information from nearby data points, KNN imputation provides a way to estimate missing values and improve the completeness of the dataset.

B. Exploratory analysis

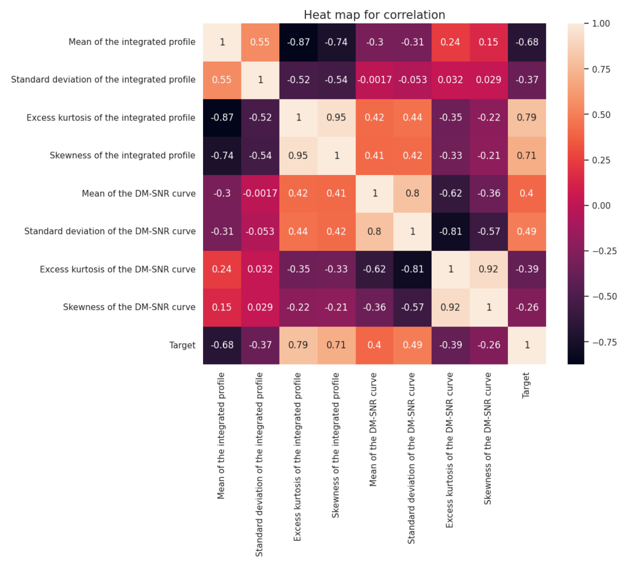


Fig. 6. Train set correlation heatmap before dropping high correlated features

1) Correlation among the numerical featurers: Two features are correlated if they have an absolute correlation coefficient close to 1. They are uncorrelated if their correlation coefficient is closer to 0. The Pearson's correlation coefficient is given as follows:

$$r_{xy} = \frac{\sum_i x_i y_i - n\bar{x}\bar{y}}{\sqrt{\sum_i x_i^2 - n\bar{x}^2}\,\sqrt{\sum_i y_i^2 - n\bar{y}^2}}.$$

The features 'skewness of integrated profile' and 'skewness of DM SNR curve' are correlated to some other features, as seen in the plot above, having a correlation coefficient greater than 0.9. So, we will drop these two features. Correlated features lead to erroneous model training. After dropping these two features, the correlation plot looks as below:

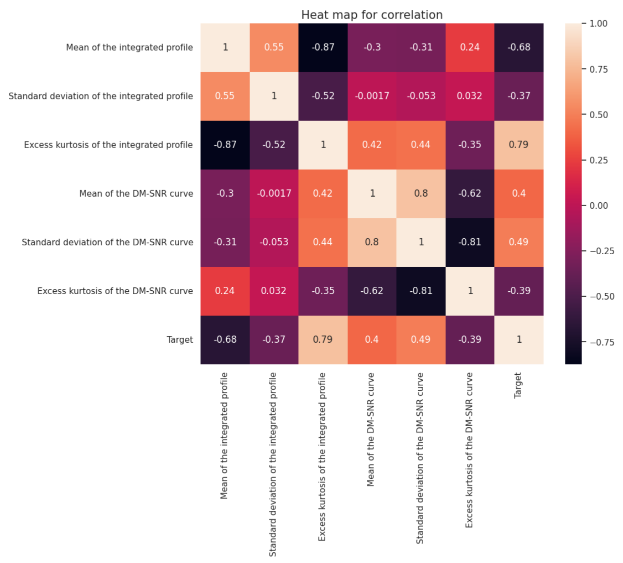


Fig. 7. Train set correlation heatmap after dropping high correlated features

2) Preliminary study on the trend followed by data: We will study a few plots before finally feeding the data to the model.

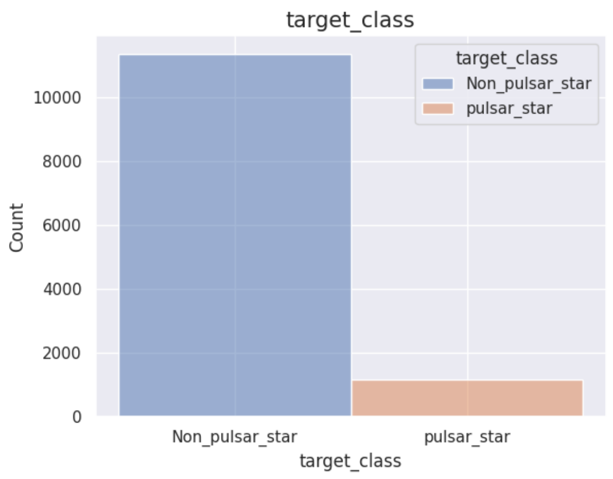


Fig. 8. Target class frequency of both classes

This plot shows that the dataset is highly imbalanced, with 11375 instances of non-pulsar stars and 1153 instances of pulsar stars.
We will look into some distribution plots before finally feeding the data into the model.

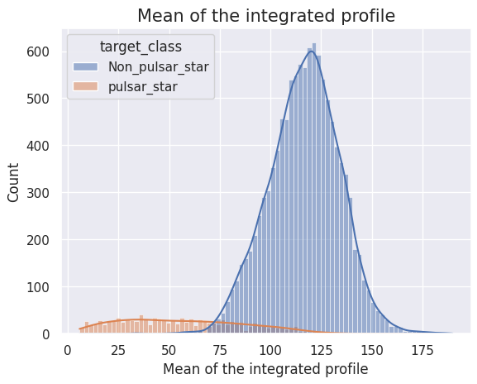


Fig. 9. Mean of the integrated profile in the train set

This is almost normally distributed from the majority class.

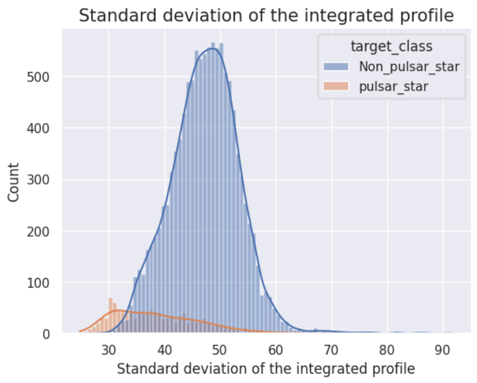


Fig. 10. Standard deviation of integrated profile in the train set

This distribution is a little left-skewed for both classes.

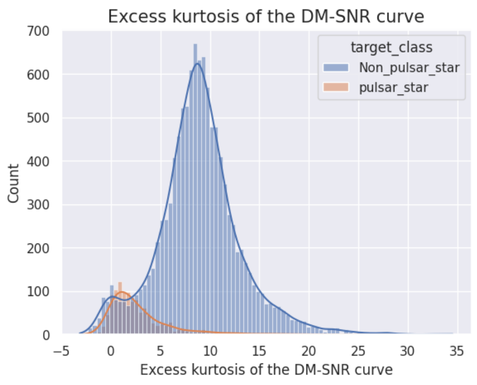


Fig. 11. Excess kurtosis of DM-SNR curve in the train set

This preliminary distribution visualization is necessary because we will crosscheck if this same distribution is followed by the test set or not later.

C. *Splitting the Training data to Train and validation sets*

The data set given to us is already split into train and test sets, with around 5000 entries in the test set. So, we need not perform a train test split again. We could have done some cross-validation, but we did not do it because the paper would get unnecessarily long.

D. *Statistical model*

Our model is a Support vector machine classifier from sklearn. Several kernels and C values were experimented of which the best kernel was found to be linear with C=10^3.

E. *Visualization and validation*

Below are the evaluation metrics for the train set:

Classes	Precision	Recall	F_1 Score
Non pulsar star	0.98	0.99	0.99
Pulsar star	0.94	0.81	0.87

TABLE II
EVALUATION METRICS

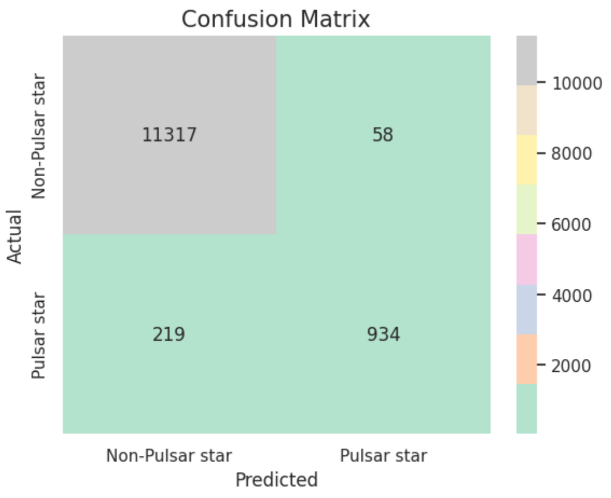


Fig. 12. Confusion Matrix with heatmap

From this figure, we can infer the following

- **True Positive**=934 (Actual Pulsar star, predicted Pulsar star)
- **False Positive**=58 (Actual Non-pulsar star, predicted pulsar star)
- **False Negetive**=219 (Actual Pulsar star, predicted Non-pulsar star)
- **True Negative**=11317 (Actual Non-pulsar star, predicted Non-pulsar star)

From the confusion matrix, we get the accuracy as:

$$\text{Accuracy} = \frac{\text{Sum of diagonal terms}}{\text{Sum of all elements}}$$

$$\text{Accuracy} = \frac{11317 + 934}{11317 + 934 + 219 + 58} = 0.9779$$

The goodness of a model can be tested by another metric called area under the ROC (Receiver operating characteristics) curve.

The more the area, the better the model is. In our case, the achieved AU-ROC is 0.97, which is fairly good.

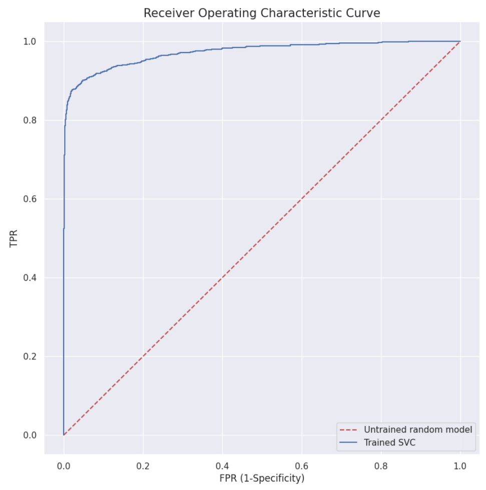


Fig. 13. Receiver operating characteristic curve

1) Insights in test data and visual validation of model:

- The relative class frequency (as predicted by the model) matches that of the train set. Pulsar star class is a minority class here also.

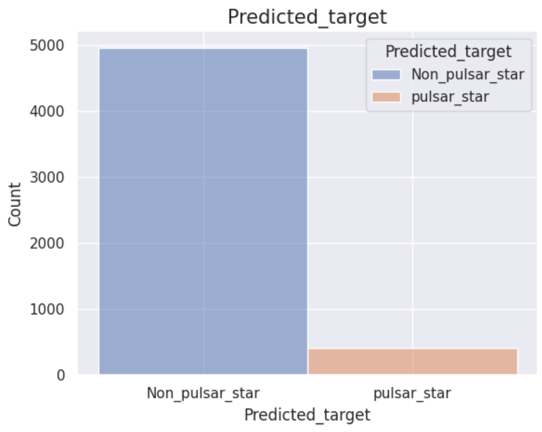


Fig. 14. Class count as predicted by the model in the test set

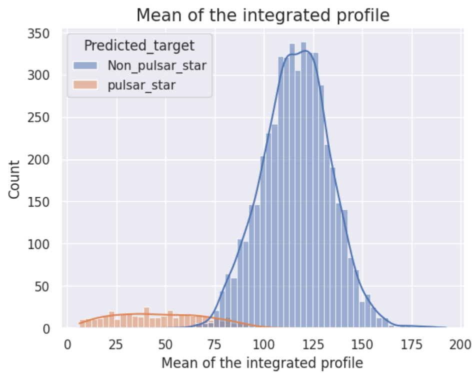


Fig. 15. Test set mean of integrated profile

- The test set mean integrated profile looks similar to the train set one. Even the labels look similar. So, the model follows the same trend as what it has seen in the train set.

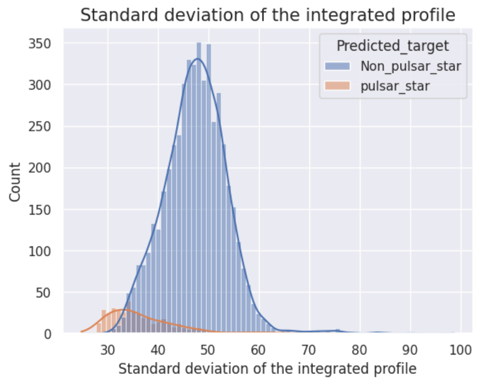


Fig. 16. Test set standard deviation of the integrated profile

- This distribution is skewed like the train set one in terms of both predicted classes.
- Excess kurtosis plot, as given below, is also quite similar to the train set.

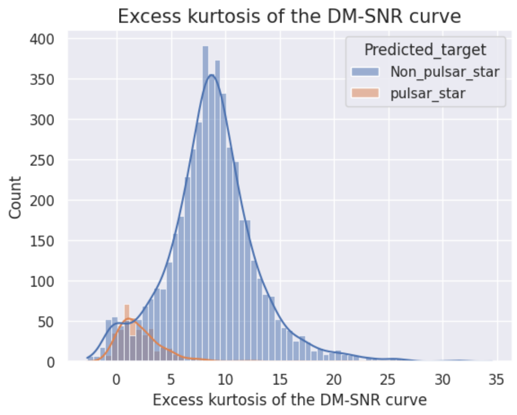


Fig. 17. DM-SNR curve excess kurtosis from the test set

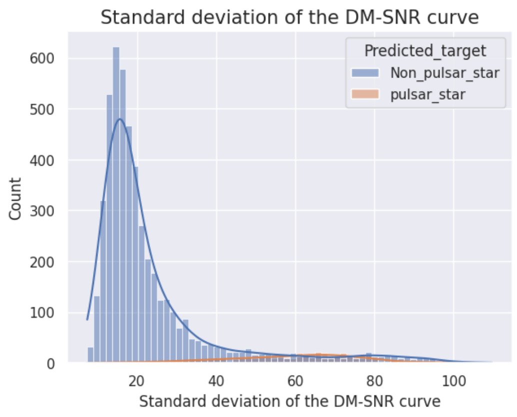


Fig. 18. Standard deviation of DM-SNR curve in the test set

So, all the distributions of the test set are similar to that of the train set. And the model behaves exactly the same way as it should.

V. Conclusions

Support vector machine is one of the most popular classification algorithms. Unlike other classification models, this model is very robust to the outliers, as it only considers a very small number of points to come up with a decision boundary. The classification capacity of this algorithm increases when the regularization parameter C is chosen properly along with the right kernel function. The primary motivation for SVM came from a very simple perceptron learning algorithm. In this assignment, we successfully demonstrated the applicability of the SVM classifier. But SVM is not only restricted to classification; it can be used in a good number of regression problems. This assignment gave us the opportunity to explore the mathematical details of SVM, introducing various mathematical tools like constrained optimization, hyperplanes, kernel functions, etc. The name is consistent with the working process of this algorithm as it uses a few data points called support vectors to construct its separating hyperplane or decision surface. This dataset allowed us to determine what visualization techniques are useful for numerical data and how to work with missing values.

References

[1] Support vector machine: https://www.geeksforgeeks.org/support-vector-machine-algorithm/

[2] Evaluation Metrics: https://www.analyticsvidhya.com/blog/2021/07/metrics-to-evaluate-your-classification-model-to-take-the-right-decisions/

[3] Pandas: https://pandas.pydata.org/

[4] Scikit learn Support Vector Machines: https://scikit-learn.org/stable/modules/svm.html

[5] Seaborn: https://seaborn.pydata.org/index.html

Access the original code here