A mathematical essay on Support Vector Machines

Assignment 6 : EE4708 Data Analytics Laboratory

Madhur Jindal
Inter Disciplinary Dual Degree Program in Data Science
Indian Institute of Technology (IIT) Madras
Chennai, India
me18b059@smail.iitm.ac.in

***Abstract*—This document is a mathematical essay on Support Vector Machines wherein it is applied on a dataset containing different attributes of a star and the key task is to classify it as a pulsar star or not, using various measures of its integrated pulse profile (folded profile). We try to see if SVMs are able to classify the dataset efficiently and which kernel is better at it, and also try to see if any feature have certain distinguishing characteristics for the both classes, such that it is very important.**

I. Introduction

Support Vector Machines (SVMs in short) are machine learning algorithms that are used for classification and regression purposes. SVMs are one of the powerful machine learning algorithms for classification, regression and outlier detection purposes. An SVM classifier builds a model that assigns new data points to one of the given categories. Thus, it can be viewed as a non-probabilistic binary linear classifier. The original SVM algorithm was developed by Vladimir N Vapnik and Alexey Ya. Chervonenkis in 1963. At that time, the algorithm was in early stages. The only possibility is to draw hyperplanes for linear classifier. In 1992, Bernhard E. Boser, Isabelle M Guyon and Vladimir N Vapnik suggested a way to create non-linear classifiers by applying the kernel trick to maximum-margin hyperplanes. The current standard was proposed by Corinna Cortes and Vapnik in 1993 and published in 1995. SVMs can be used for linear classification purposes. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using the kernel trick. It enable us to implicitly map the inputs into high dimensional feature spaces.

II. Support Vector Machines

A. *Intuition*

Hyperplane

A hyperplane is a decision boundary which separates between given set of data points having different class labels. The SVM classifier separates data points using a hyperplane with the maximum amount of margin. This hyperplane is known as the maximum margin hyperplane and the linear classifier it defines is known as the maximum margin classifier.

Support Vectors

Support vectors are the sample data points, which are closest to the hyperplane. These data points will define the separating line or hyperplane better by calculating margins.

Margin

A margin is a separation gap between the two lines on the closest data points. It is calculated as the perpendicular distance from the line to support vectors or closest data points. In SVMs, we try to maximize this separation gap so that we get maximum margin.

The following diagram illustrates these concepts visually.

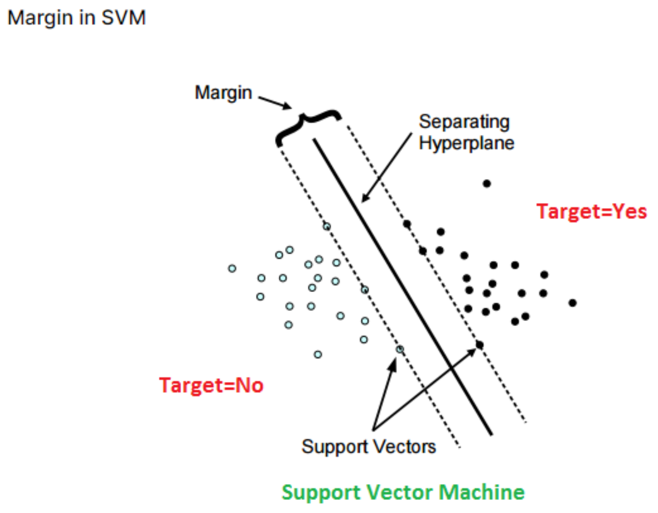


Fig. 1.

B. *Under the Hood*

In SVMs, our main objective is 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 –

Generate hyperplanes which segregates the classes in the best possible way. There are many hyperplanes that might classify the data. We should look for the best hyperplane that represents the largest separation, or margin, between the two classes.

So, we choose the hyperplane so that distance from it to the support vectors on each side is maximized. If such

Identify applicable funding agency here. If none, delete this.

a hyperplane exists, it is known as the maximum margin hyperplane and the linear classifier it defines is known as a maximum margin classifier.

The following diagram illustrates the concept of maximum margin and maximum margin hyperplane in a clear manner.

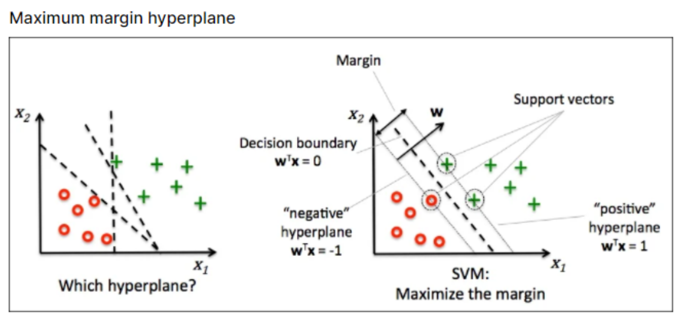


Fig. 2.

C. Problem with dispersed datasets

Sometimes, the sample data points are so dispersed that it is not possible to separate them using a linear hyperplane. In such a situation, SVMs uses a kernel trick to transform the input space to a higher dimensional space as shown in the diagram below. 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.

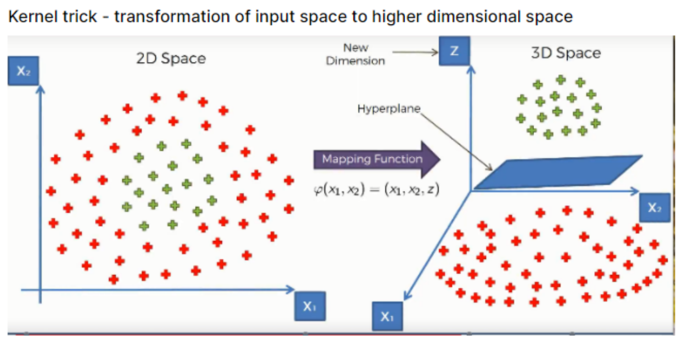

Kernel trick - transformation of input space to higher dimensional space
2D Space
New Dimension
Z
3D Space
Hyperplane
Mapping Function
$\varphi(x_1, x_2) = (x_1, x_2, z)$
X₁
X₂

Fig. 3.

D. Kernel Trick

In practice, SVM algorithm is implemented using a kernel. It uses a technique called the kernel trick. In simple words, 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 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-

In the context of SVMs, there are 4 popular kernels – Linear kernel,Polynomial kernel,Radial Basis Function (RBF) kernel

Kernel function

$$K(\bar{x}) = \begin{cases} 1 & \text{if } \|\bar{x}\| \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

Fig. 4.

(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 γ parameter. So, performing a grid search will usually take more time.

Linear kernel can be visualized with the following figure.

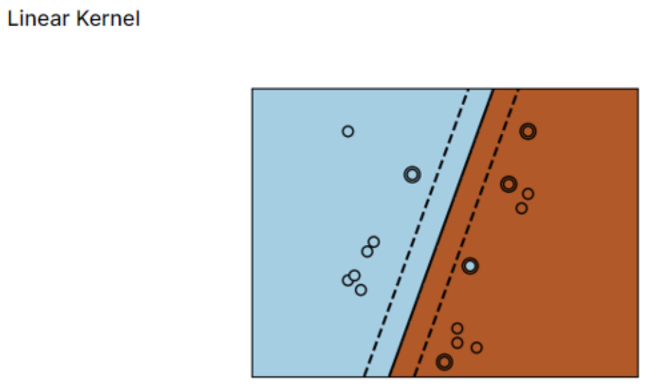


Fig. 5.

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 combinations of the input samples.

For degree-d polynomials, the polynomial kernel is defined as follows –

Polynomial kernel :

$$K(x_i, x_j) = (\gamma x_i^T x_j + r)d, \gamma > 0$$

Polynomial kernel is very popular in Natural Language Processing. The most common degree is d = 2 (quadratic), since larger degrees tend to overfit on NLP problems. It can be visualized with the following diagram.

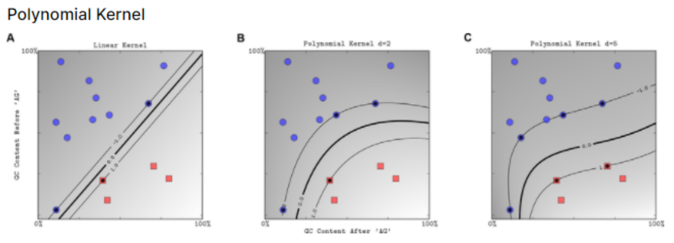


Fig. 6.

E. 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 –

Radial Basis Function kernel

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

Fig. 7.

The following diagram demonstrates the SVM classification with rbf kernel.

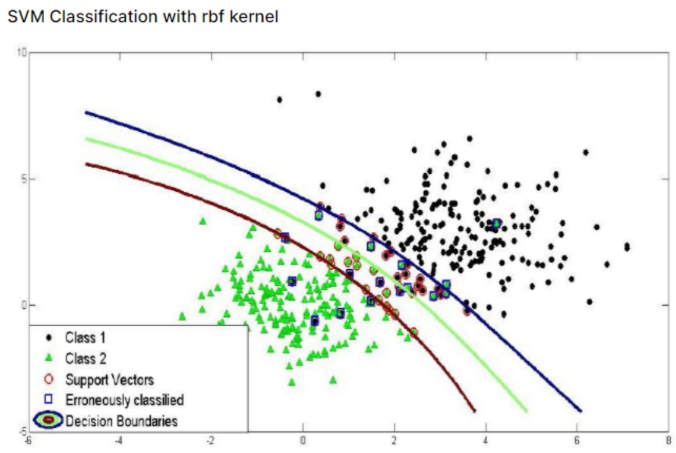


Fig. 8.

1) Sigmoid Kernel: Sigmoid kernel has its origin in neural networks. We can use it as the proxy for neural networks. Sigmoid kernel is given by the following equation –

sigmoid kernel :

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

Sigmoid kernel can be visualized with the following diagram-

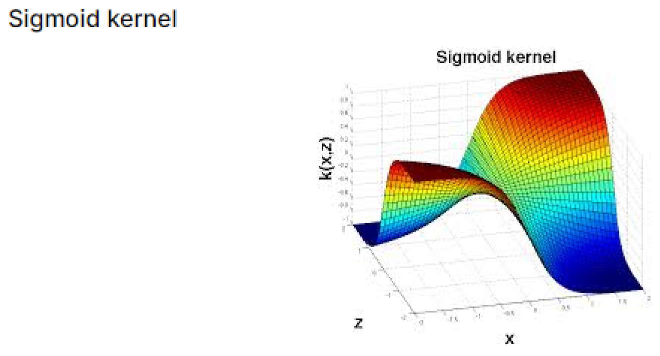


Fig. 9.

F. SVM Scikit-Learn libraries

Scikit-Learn provides useful libraries to implement Support Vector Machine algorithm on a dataset. There are many libraries that can help us to implement SVM smoothly. We just need to call the library with parameters that suit to our needs. In this project, I am dealing with a classification task. So, I will mention the Scikit-Learn libraries for SVM classification purposes.

First, there is a LinearSVC() classifier. As the name suggests, this classifier uses only linear kernel. In LinearSVC() classifier, we don't pass the value of kernel since it is used only for linear classification purposes.

Scikit-Learn provides two other classifiers - SVC() and NuSVC() which are used for classification purposes. These classifiers are mostly similar with some difference in parameters. NuSVC() is similar to SVC() but uses a parameter to control the number of support vectors. We pass the values of kernel, gamma and C along with other parameters. By default kernel parameter uses rbf as its value but we can pass values like poly, linear, sigmoid or callable function.

G. Accuracy Metrics

		Prediction outcome		
		p	**n**	**total**
actual value	**p′**	True Positive	False Negative	P′
	n′	False Positive	True Negative	N′
	total	P	N	

we can assign the event row as "positive" and the no-event row as "negative". We can then assign the event column of predictions as "true" and the no-event as "false".

This gives us:

"true positive" for correctly predicted event values. "false positive" for incorrectly predicted event values. "true negative" for correctly predicted no-event values. "false negative" for incorrectly predicted no-event values.

1) Precision: Precision can be defined as the percentage of correctly predicted positive outcomes out of all the predicted positive outcomes. It can be given as the ratio of true positives (TP) to the sum of true and false positives (TP + FP).

So, Precision identifies the proportion of correctly predicted positive outcome. It is more concerned with the positive class than the negative class.

Mathematically, precision can be defined as the ratio of TP to (TP + FP).

2) Recall: Recall can be defined as the percentage of correctly predicted positive outcomes out of all the actual positive outcomes. It can be given as the ratio of true positives (TP) to the sum of true positives and false negatives (TP + FN). Recall is also called Sensitivity.

Recall identifies the proportion of correctly predicted actual positives.

Mathematically, recall can be given as the ratio of TP to (TP + FN).

3) F1-Score: f1-score is the weighted harmonic mean of precision and recall. The best possible f1-score would be 1.0 and the worst would be 0.0. f1-score is the harmonic mean of precision and recall. So, f1-score is always lower than accuracy measures as they embed precision and recall into their computation. The weighted average of f1-score should be used to compare classifier models, not global accuracy.

4) Support: Support is the actual number of occurrences of the class in our dataset.

5) Null-Hypothesis and P-value: The p-value for each term tests the null hypothesis that the coefficient is equal to zero (no effect). A low p-value (¡ 0.05) indicates that you can reject the null hypothesis. In other words, a predictor that has a low p-value is likely to be a meaningful addition to your model because changes in the predictor's value are related to changes in the response variable.

Conversely, a larger (insignificant) p-value suggests that changes in the predictor are not associated with changes in the response.

III. Problem - Understanding and Modelling

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. Classification algorithms in particular are being adopted, which treat the data sets as binary classification problems. Here the legitimate pulsar examples form minority positive class and spurious examples form the majority negative class.

The data set shared here contains 11376 negative examples caused by RFI/noise, and 1,152 real pulsar examples. Each row lists the variables first, and the class label is the final entry. The class labels used are 0 (negative) and 1 (positive). Each candidate is described by 8 continuous variables, and a single class variable. The first four are simple statistics obtained from the integrated pulse profile. The remaining four variables are similarly obtained from the DM-SNR curve . These are summarised below:

1. Mean of the integrated profile.
2. Standard deviation of the integrated profile.
3. Excess kurtosis of the integrated profile.
4. Skewness of the integrated profile.
5. Mean of the DM-SNR curve.
6. Standard deviation of the DM-SNR curve.
7. Excess kurtosis of the DM-SNR curve.
8. Skewness of the DM-SNR curve.
9. Class

A. Data Pre-processing

We start with reading the data to a pandas dataframe. We observe a total of 12528 data points, with 9 columns in total, On visualizing the distributions of the target variable, we see an imbalanced dataset problem, wherein around 9.2% of the total stars cars are classified as pulsars, 90.8% are not (Figure 1). We see that all the 6 features are numerical. We then move on to checking for the null values in the data and find that three columns have null values in both the train and test datasets, which are Excess kurtosis of the integrated profile, Standard deviation of the DM-SNR curve and Skewness of the DM-SNR curve. Thus, we need to treat these. Hence we compute the median values for each of these columns and impute the missing values in both the training and test sets as these median values, as median values are much less likely to be affected by outliers.

B. Exploratory Data Analysis

We start with a univariate analysis, wherein we create histplots for each of the eight features with hue as the target column, using the seaborn library. We see that for features as Mean, standard deviation, kurtosis and skewness of integrated

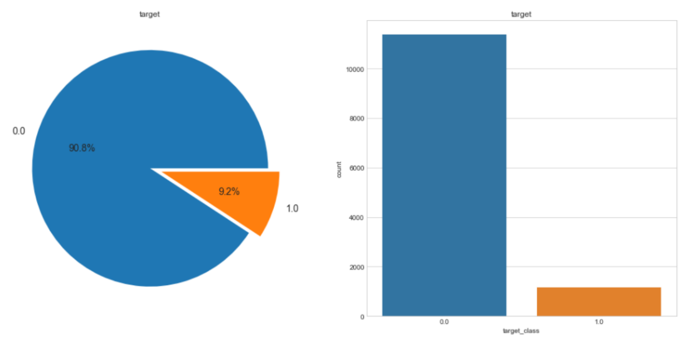


Fig. 10. Target Distribution

profile, the standard deviation is quite high, that is the values are distributed much more in the positive class and are concentrated in the negative class. For the remaining features we see an opposite trend that is the values are distributed much more in the class 0 and are concentrated in the 1 class. We also see that for features like mean, standard deviation of integrated profile and kurtosis, skewness of the DM-SNR curve, the mean is lower in the positive class than the negative class, while it is the opposite for the other features.

Moving on to bivariate analysis, wherein we create pairpolots for each of the columns we see that all the features have a clear boundary of separation and hence we would be able to perform well with our SVM model. We only fee some overlap near the decision boundary regions which can be tuned using regularization values (C values) to find a good fit.

We then move on to creating a correlation map for the dataset and see high correlation between the values, which can cause trouble and should be handled if our accuracy or f1 values are not good enough. Some of the methods can be removing some features or creating hybrid combinations of the features.

C. Postprocessing

We split the data using a 80/20 split, with the final dataset being 10022 examples in training set and 2506 examples in crossvalidation set. We then use the Standard scaler to scale our train and validation values.

D. Support Vector Machine Modelling

We start with modelling using the default hyperparameters usnig the SVC library of sklearn. Default hyperparameter means C=1.0, kernel=rbf and gamma=auto among other parameters.We observe model accuracy score with default hyperparameters: 0.9741 and model f1 score with default hyperparameters: 0.8426.Based on the above analysis we can conclude that our classification model accuracy is very good. Our model is doing a very good job in terms of predicting the class labels. But, this is not true. Here, we have an imbalanced dataset. The problem is that accuracy is an inadequate measure for quantifying predictive performance in the imbalanced dataset problem. So, we must explore alternative metrices that provide better guidance in selecting models. In particular, we would like to know the underlying distribution of values and the type of errors our classifer is making. Thus we check for the f1 score which is helpful with such imbalanced datasets. We see that our model is performing well on the f1 score with a score of 0.84.

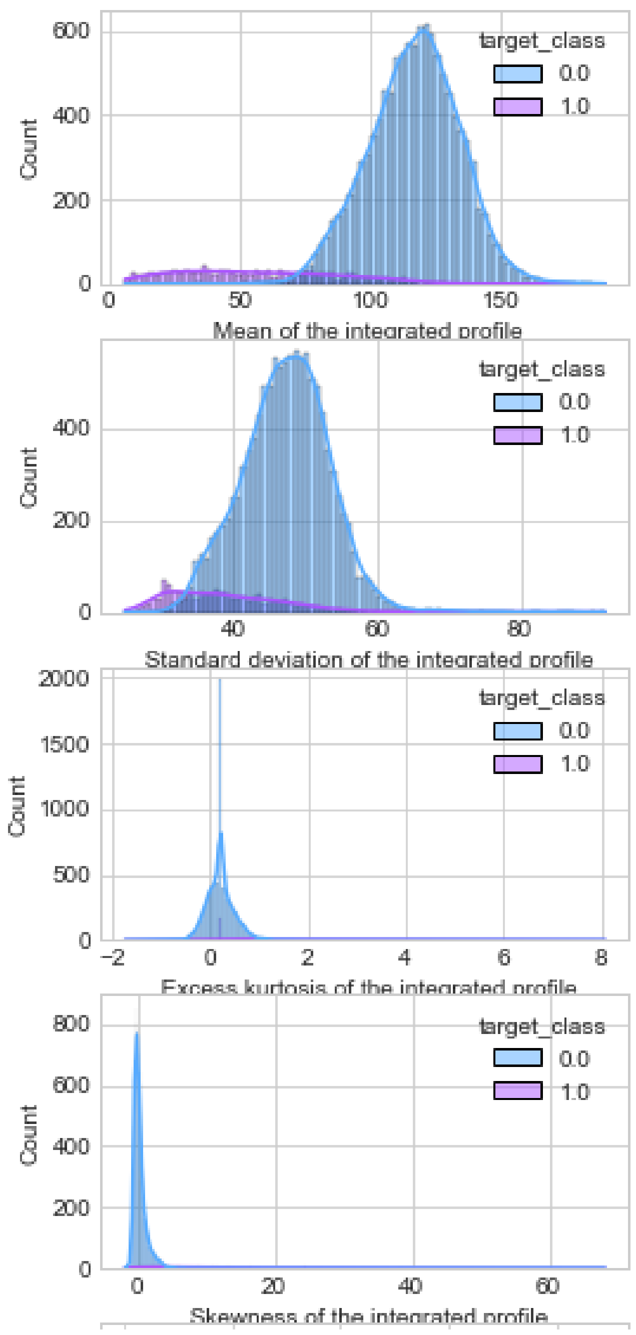


Fig. 11. Univariate Analysis

But we are not done yeat. We can tune the hyperparameters to get a better score and a better fit. Thus we use Grid Search to search from our defined space of hyperparameters such as trying the different kernels, with C ranging from 0.01 to 10. We also vary the degree for the linear kernel

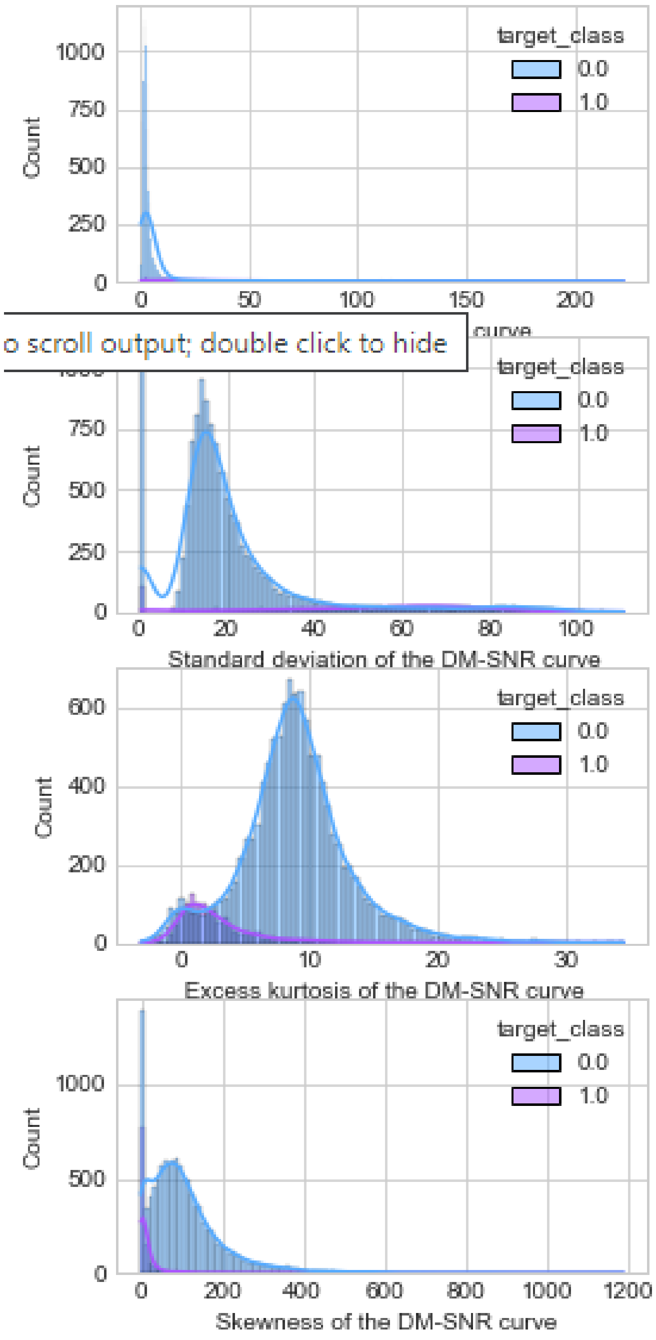


Fig. 12. Univariate Analysis

and set the class weightsa s balanced. We also check for the gamma hyperparameter with values as scale and auto. Finally using a 2 fold cross validation we get the best model with 0.87 F1 score on the train and 0.85 on the validation which is an improvement over the initial score of 0.84. The best hyperparameters that are chosen are rbf kernel with auto gamma and C = 2.51 which is higher than the default of 1. Thus we are reducing the regularization.

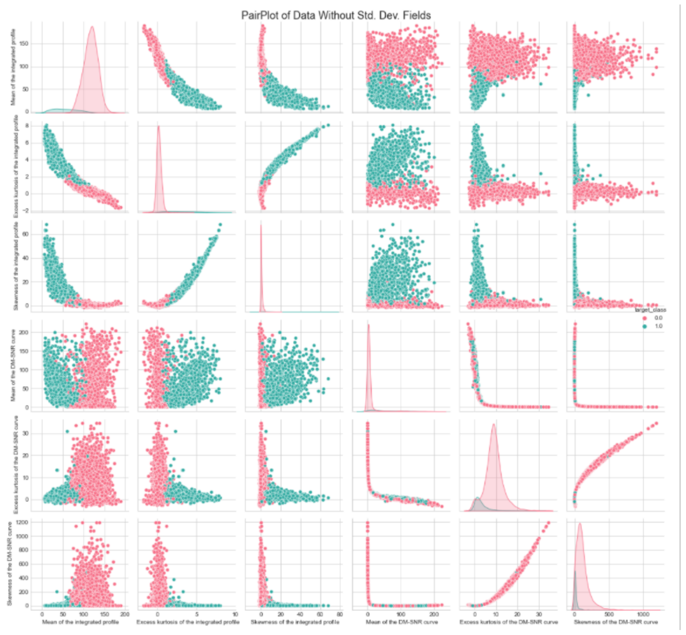
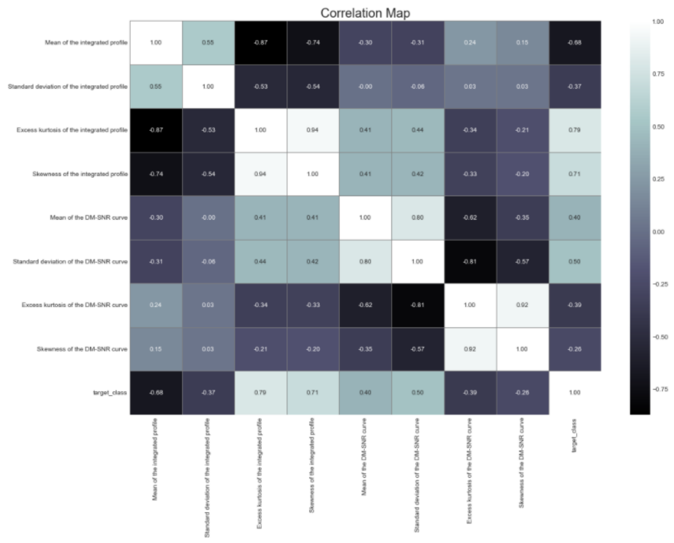


Fig. 13. Multivariate Analysis



	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
Mean of the integrated profile	1.00	0.55	-0.87	-0.74	-0.30	-0.31	0.24	0.15	-0.68
Standard deviation of the integrated profile	0.55	1.00	-0.53	-0.54	-0.00	-0.06	0.03	0.03	-0.37
Excess kurtosis of the integrated profile	-0.87	-0.53	1.00	0.94	0.41	0.44	-0.34	-0.21	0.79
Skewness of the integrated profile	-0.74	-0.54	0.94	1.00	0.41	0.42	-0.33	-0.20	0.71
Mean of the DM-SNR curve	-0.30	-0.00	0.41	0.41	1.00	0.80	-0.62	-0.35	0.40
Standard deviation of the DM-SNR curve	-0.31	-0.06	0.44	0.42	0.80	1.00	-0.81	-0.57	0.50
Excess kurtosis of the DM-SNR curve	0.24	0.03	-0.34	-0.33	-0.62	-0.81	1.00	0.92	-0.39
Skewness of the DM-SNR curve	0.15	0.03	-0.21	-0.20	-0.35	-0.57	0.92	1.00	-0.26
target_class	-0.68	-0.37	0.79	0.71	0.40	0.50	-0.39	-0.26	1.00

Fig. 14. Correlation Map

IV. Conclusion

Having done a thorough analysis of the Support Vector methods with different kernels, we get an imporvement by 0.01 in the F1 score using GridSearchCV. So, GridSearch CV helps to identify the parameters that will improve the performance for this particular model.There are outliers in our dataset. So, as I increase the value of C to limit fewer outliers, the accuracy increased. This is true with different kinds of kernels. ROC AUC of our model is very close to 1. So, we can conclude that our classifier does a good job in classifying the pulsar star. We also get good precision and recall values of 0.98 and 0.985 respectively. The true positive rate values is 0.985 while the False Positive rate is at 0.191.

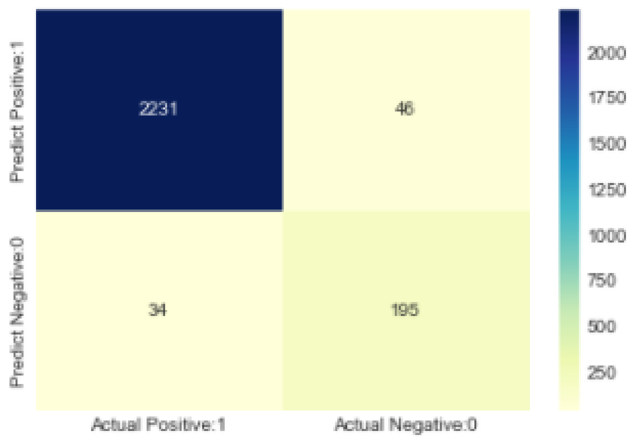


Fig. 15. Confusion Matrix

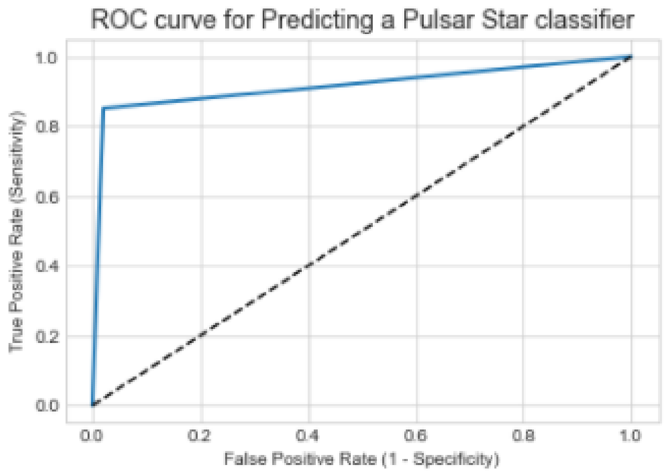


Fig. 16.

We conclude that SVMs with rbf kernel are able to get a good fit on our training data as expected using the multivariate analysis where we found that most of the features divide the target classes into two regions which are very separable and prominent. Further avenues of growth can be checking for more features that could explain the target variable better. We could also try reducing the correlation in the features by eliminating some features or creating hybrid features. As the data is very imbalanced we could also try using upsampling and downsampling techniques.

References

[1] Support Vector Machines, Data Analytics Laboratory EE4708, July - November 2021

[2] Support Vector Machines by Prof. Manikandan, Pattern Recognition and Machine Learning, July - November 2021

[3] https://en.wikipedia.org/wiki/Support-vector_machine

[4] https://www.datacamp.com/community/tutorials/svm-classification-scikit-learn-python

[5] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

[6] http://dataaspirant.com/2017/01/13/support-vector-machine-algorithm/

[7] https://www.kaggle.com/prashant111/svm-classifier-tutorial

[8] https://www.ritchieng.com/machine-learning-evaluate-classification-model/

[9] https://en.wikipedia.org/wiki/Kernel_method

[10] https://en.wikipedia.org/wiki/Polynomial_kernel

[11] https://en.wikipedia.org/wiki/Radial_basis_function_kernel