Using Support Vector Machines to Predict Pulsar Stars

Swathi V
Dept. of Engineering Design
Indian Institute of Technology Madras
Chennai, India
ed18b034@smail.iitm.ac.in

***Abstract*—Support vector machines are versatile supervised machine learning frameworks, which can be used for two-class classification problems. In this report, we look at the mathematics behind support vector machines, and use the same to build a predictive model for classifying stars as pulsar stars or otherwise. We also look at how their performance can be quantified, and what parameters can be tuned to improve the performance.**

***Index Terms*—Maximum margin classifier, support vector classifier, support vector machines, pulsar stars**

I. Introduction

Support vector machines are one of the classical machine learning algorithms that can be used for classification purposes. It is a supervised learning algorithm that has proven to give good results in many applications, especially involving big data.

Basic support vector machines work by finding the optimal hyperplane to separate the given data points into two classes. However, this approach can be generalised to accommodate data that is not linearly separable as well, using soft margin classifiers or kernels. In this report, we look at how these classifiers are formulated, and how the optimal classifier is chosen.

Pulsars are a type of Neutron star that produce radio emissions that can be detected on earth. They are of considerable scientific interest, as probes of space-time, the inter-stellar medium, and states of matter. The aim of this assignment is to use support vector machines to try and classify various stars as pulsars or otherwise.

In this report, we start by understanding the mathematics behind support vector machines, starting from maximum margin classifiers and generalising them further to support vector classifiers and SVMs subsequently. We then analyse the dataset and try to build a predictive model using SVM models, which we then use to further understand the dataset and the dependency between variables.

II. Support Vector Machines - A Mathematical Overview

The support vector machine is a generalization of an intuitive classifier caller the **maximum margin classifier**. This intuitive and elegant classifier, however, can only be applied to linearly separable datasets. In this section, we start with the maximum margin classifier, and then introduce the **support vector classifier**, which is an extension that can be applied to a broader set of data. We then look at **support vector machines**, which can accommodate data having non-linear classification boundaries as well.

A. Maximum Margin Classifier

The maximum margin classifier works by constructing a separating hyperplane that separates the two classes in the data. However, if such a hyperplane exists, then there will exist an infinite number of such hyperplanes (which can be obtained by translating or rotating the original hyperplane by a small amount).

An intuitive choice for the hyperplane is the one that is farthest from all the training observations, also known as the *maximum margin hyperplane* or the *optimal separating hyperplane*. The smallest distance between the hyperplane and any of the training data points is called the *margin*. Thus, the maximum margin hyperplane is one which is farthest away from the data points. The data points which are closest to the hyperplane, and thus decide the width of the margin, are called *support vectors*. We can then classify any new data point depending on which side of the hyperplane it falls on.

Suppose we have n data points for training, each in p dimensional space. Let the output classes be given by $y_i = 1$ or $y_i = -1$. Let the separating hyperplane be given by

$$\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p = 0 \tag{1}$$

It then has the properties

$$\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} > 0 \text{ if } y_i = 1 \tag{2}$$

and

$$\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} < 0 \text{ if } y_i = 1. \tag{3}$$

Equivalently, a separating hyperplane has the property that

$$y_i \left(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}\right) > 0 \; \forall \; i = 1, 2, \cdots, n \tag{4}$$

The maximum margin classifier can be found by solving the following optimization problem:

$$\max_{\beta_0,\beta_1,\cdots,\beta_p} M \tag{5}$$

$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1 \tag{6}$$

$$\text{and } y_i\left(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}\right) \geq M \;\forall\; i = 1, 2, \cdots, n \tag{7}$$

$$\text{where } M \geq 0. \tag{8}$$

The constraints ensure that all observations are on the correct side of the hyperplane, and the perpendicular distance of an observation from the hyperplane is given by

$$y_i\left(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}\right). \tag{9}$$

However, as mentioned earlier, the maximum margin classifier works only for linearly separable data. If no separating hyperplane exists, the optimization problem has no solution for $M > 0$. To handle such cases, we can extend the maximum margin classifier to a *soft margin classifier*, or a support vector classifier.

B. Support Vector Classifier

A soft-margin classifier is sometimes more desirable than a maximum margin classifier, in the interest of better robustness and generalization. The soft-margin classifier allows a few data points to be on the wrong side of the hyperplane, which is inevitable if the data is not linearly separable.

The hyperplane is found by solving the optimization problem:

$$\max_{\beta_0,\beta_1,\cdots,\beta_p,\epsilon_1,\cdots,\epsilon_n} M \tag{10}$$

$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1 \tag{11}$$

$$\text{and } y_i\left(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}\right) \geq M(1-\epsilon_i) \;\forall\; i \tag{12}$$

$$\text{where } M \geq 0, \epsilon_i \geq 0 \;\forall\; i, \text{ and } \sum_{i=1}^{n} \epsilon_i \leq C. \tag{13}$$

Here, C is tuning parameter that controls the level of slack allowed, and $\epsilon_1, \cdots, \epsilon_n$ are the slack variables. For any given training point x_i, if $\epsilon_i = 0$, then the point is on the correct side of the margin, if $\epsilon_i > 0$, then it is on the wrong side of the margin, and if $\epsilon_i > 1$, the point is on the wrong side of the hyperplane. Since $\sum_{i=1}^{n} \epsilon_i \leq C$, a higher value of C indicates more slack, or that more variable are allowed to be on the wrong side of the hyperplane. The parameter C is tuned using cross-validation or validation settings, keeping in mind the bias-variance tradeoff.

Though the support vector classifier allows for some slack in the decision boundaries, they are not applicable in cases where the boundaries are not linear. In such cases, we generalize the SVC using *kernels*, to construct a *Support Vector Machine*.

C. Support Vector Machines

The linear support vector classifier described above can be represented as

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle \tag{14}$$

where $\langle x, x_i \rangle$ represents the inner product of the new data point x with the training data point x_i. The inner product is given by

$$\langle x_1, x_2 \rangle = \sum_{j=1}^{p} x_{1j} x_{2j} \tag{15}$$

The n parameters $\alpha_1, \alpha_2, \cdots \alpha_n$ represent the weights of each training data point. However, α_i is non-zero only if x_i is a support vector, i.e., lies on or inside the margin in a support vector classifier. We can thus rewrite the linear support vector classifier as

$$f(x) = \beta_0 + \sum_{i:x_i \in S} \alpha_i \langle x, x_i \rangle \tag{16}$$

where S is the set of support vectors.

In order to generalize the support vector classifier to include non-linear boundaries, we replace the inner product with the *kernel function*, represented by $K(x, x_i)$. The kernel function is any function that quantifies the similarity between the two data points. The support vector machine is then given by the equation

$$f(x) = \beta_0 + \sum_{i:x_i \in S} \alpha_i K(x, x_i) \tag{17}$$

The common choices for kernel functions are listed below.

1) Linear kernel: The linear kernel function is simply the inner product of the two data points, using which we can get the support vector classifier.

2) Polynomial kernel: A polynomial kernel of degree d (a positive integer) is given by

$$K(x_1, x_2) = \left(1 + \sum_{j=1}^{d} x_{1j} x_{2j}\right)^d \tag{18}$$

which leads to a more flexible boundary than a linear kernel.

3) Radial Kernel: The radial kernel is given by

$$K(x_1, x_2) = \exp\left(-\gamma \sum_{j=1}^{p} (x_{1j} - x_{2j})^2\right) \tag{19}$$

where γ is a positive constant.

The classification is done by computing $f(x)$ for a new data point, and depending on the sign of the result, the data point is classified either as $y = 1$ or $y = -1$.

III. Data

The key aim of this report is to predict if a star is a pulsar star (1) or not (0), based on the following variables:

- 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

A. Preliminary Data Analytics

The train dataset consists of **12528** data points, and the test dataset consists of **5370** data points. The split of the train dataset with respect to the target classes is shown in Fig. 1.

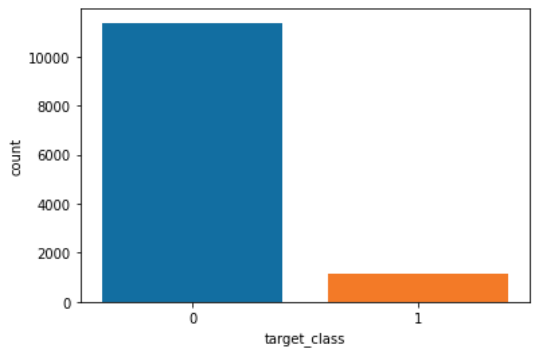


Fig. 1. Distribution of the target variable

IV. The Problem

For the given dataset, we try to predict the target variable using the given input variables. The steps followed for the same are:

- Data cleaning
- Data Visualization and Analysis
- Building SVM classifiers with various kernels and choosing the best one
- Using the model to understand the most important features
- Using the model to make predictions on the test data

A. Data Cleaning

There are missing values in both the train and test dataset, which have been replaced using class-wise *mean imputation* in the train dataset. For the test dataset, since the class data is not available, we use mean imputation on the whole dataset.

In addition, the train dataset is extremely skewed, as can be seen from Fig. 1. Hence, if the results are not satisfactory, dataset augmentation can be used to further improve the results.

B. Data Visualization and Analysis

To visualise the distribution of each input feature, a histogram was plotted for each input feature, colour coded by the 'target' variable. The results are shown in Fig. 2 to Fig. 9.

From these figures, we see that variables 'skew_int' and 'mean_dm_snr' seem to be distributed equally for both classes. The rest of the variables (especially ones like *ex_kurtosis_int*) show very distinct histograms from the two classes.

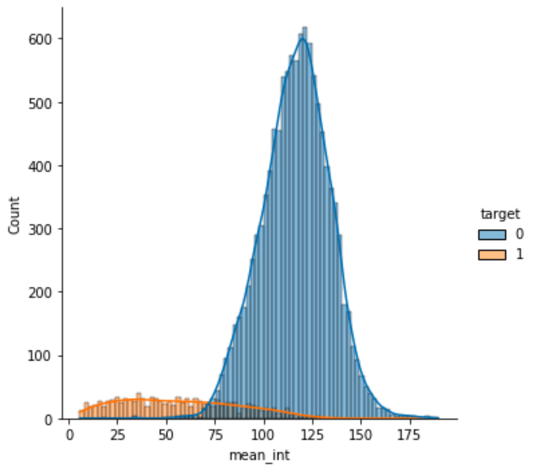


Fig. 2. Distribution of *mean_int* for both classes

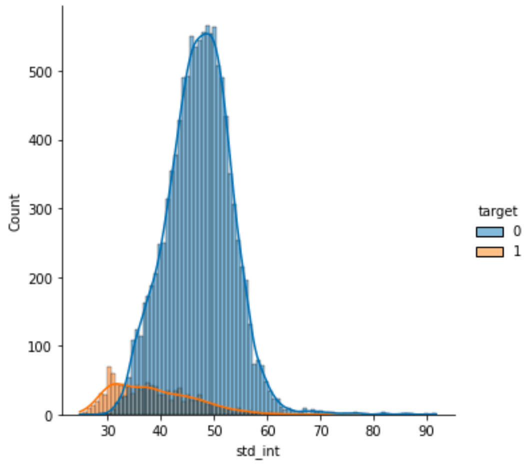


Fig. 3. Distribution of *std_int* for both classes

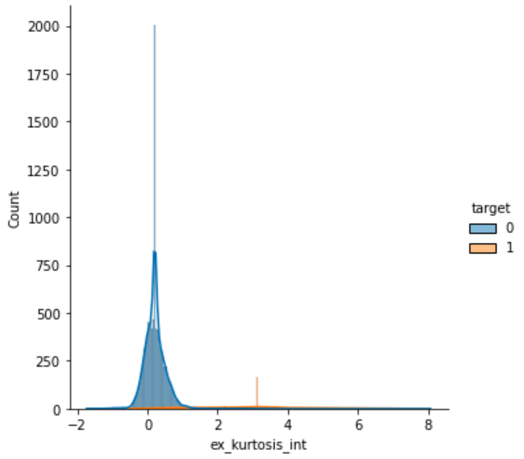


Fig. 4. Distribution of *ex_kurtosis_int* for both classes

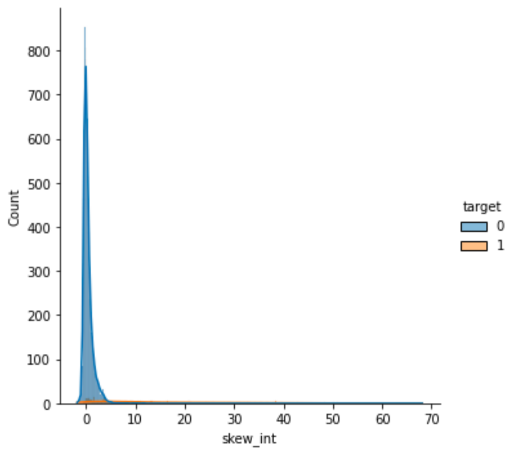


Fig. 5. Distribution of *skew_int* for both classes

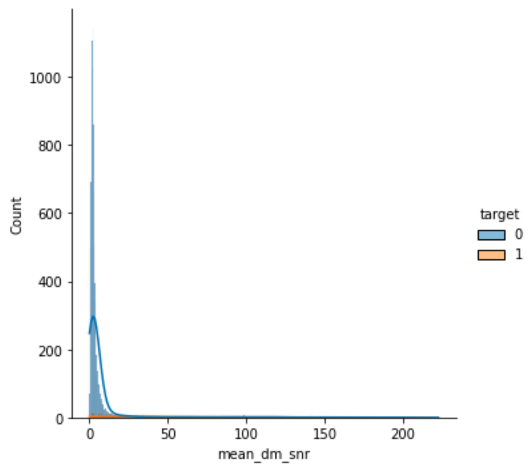


Fig. 6. Distribution of *mean_dm_snr* for both classes

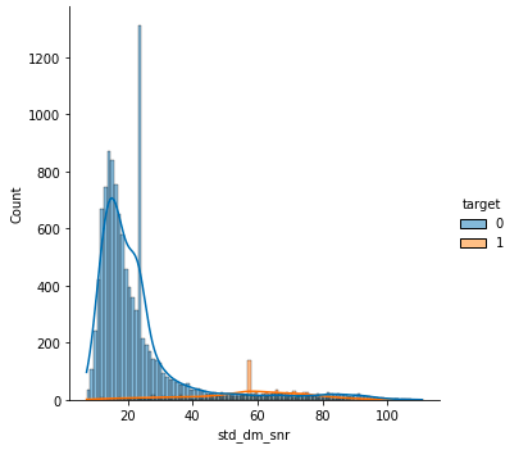


Fig. 7. Distribution of *std_dm_snr* for both classes

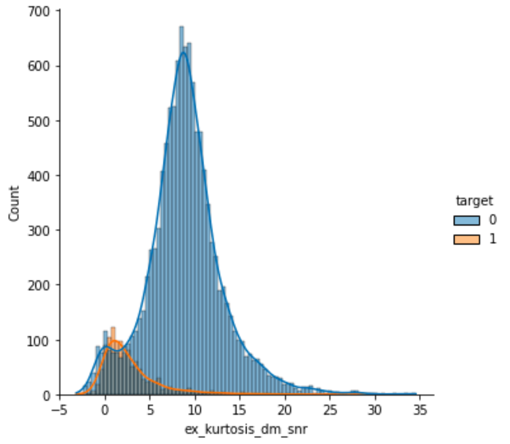


Fig. 8. Distribution of *ex_kurtosis_dm_snr* for both classes

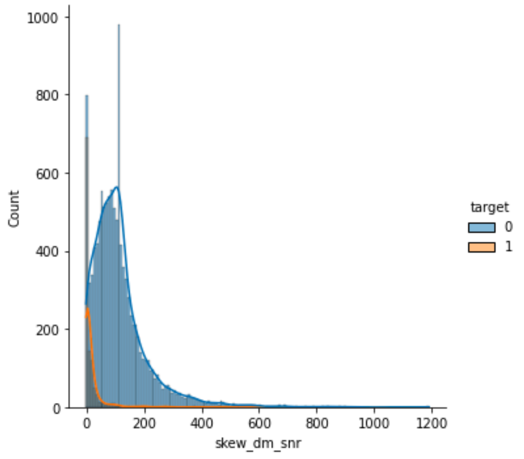
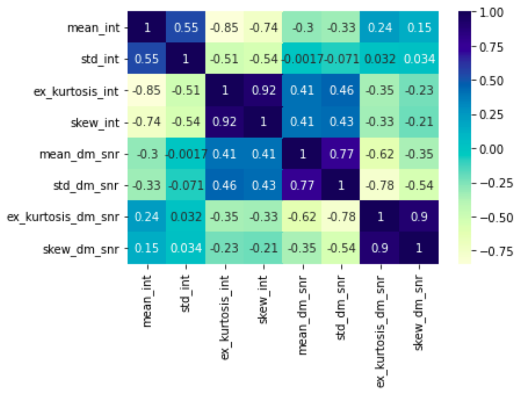


Fig. 9. Distribution of *skew_dm_snr* for both classes

In addition to this, we also plot the correlation heatmap between pairs of input variables, which would aid in feature selection. The result is shown in Fig. 10.



	mean_int	std_int	ex_kurtosis_int	skew_int	mean_dm_snr	std_dm_snr	ex_kurtosis_dm_snr	skew_dm_snr
mean_int	1	0.55	-0.85	-0.74	-0.3	-0.33	0.24	0.15
std_int	0.55	1	-0.51	-0.54	-0.0017	-0.071	0.032	0.034
ex_kurtosis_int	-0.85	-0.51	1	0.92	0.41	0.46	-0.35	-0.23
skew_int	-0.74	-0.54	0.92	1	0.41	0.43	-0.33	-0.21
mean_dm_snr	-0.3	-0.0017	0.41	0.41	1	0.77	-0.62	-0.35
std_dm_snr	-0.33	-0.071	0.46	0.43	0.77	1	-0.78	-0.54
ex_kurtosis_dm_snr	0.24	0.032	-0.35	-0.33	-0.62	-0.78	1	0.9
skew_dm_snr	0.15	0.034	-0.23	-0.21	-0.35	-0.54	0.9	1

Fig. 10. Correlation between pairs of input variables

C. Building SVM classifiers with various kernels and choosing the best one

SVMs using *linear* and *rbf* kernels were used, the results of which are shown in Fig. 11 to Fig. 15. From the above said graphs, we see that a *linear* kernel fits the data better than an *RBF* kernel, and generalises well to the training and test data. Thus, our final model is trained using a linear kernel, and the ROC curve for the same is shown in Fig. 16.

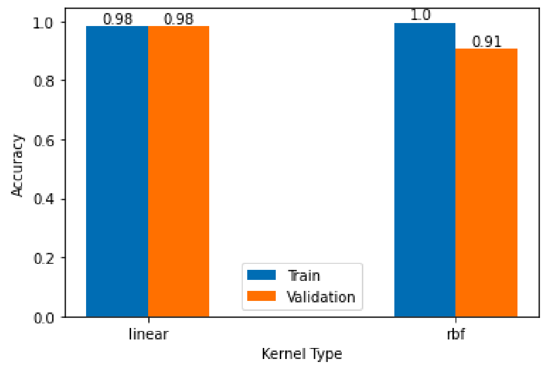


Fig. 11. Accuracy of various models

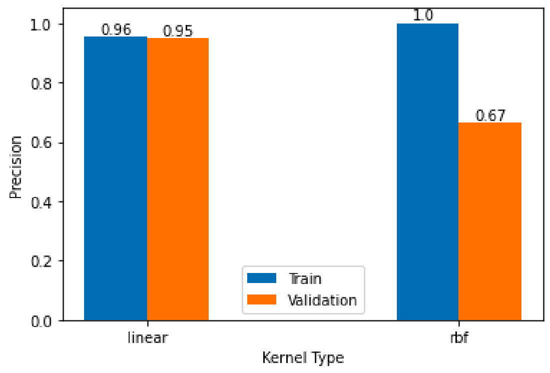


Fig. 12. Precision of various models

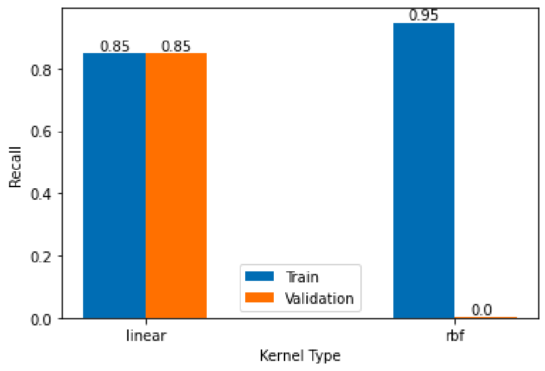


Fig. 13. Recall of various models

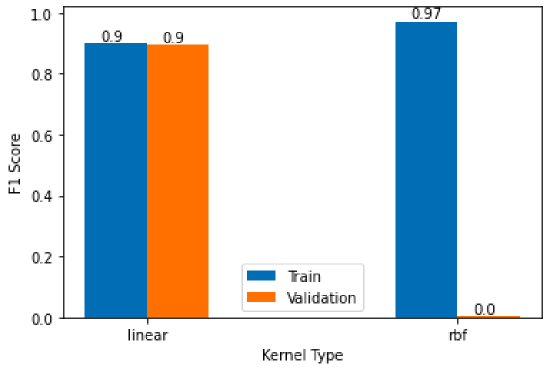


Fig. 14. F1 score of various models

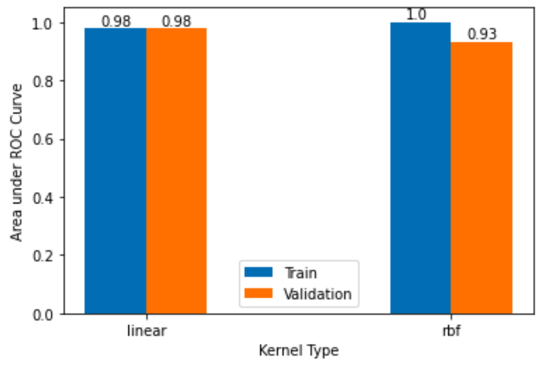


Fig. 15. Area under ROC curve of various models

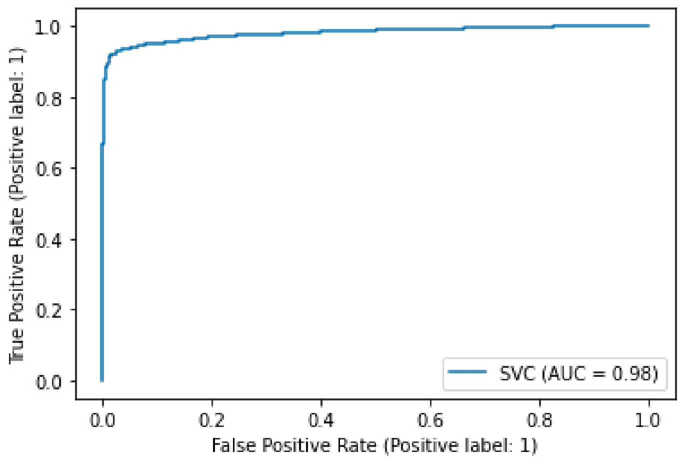


Fig. 16. ROC curve for the model

D. Using the model to understand the most important features

Since the model chosen is linear, the coefficients of the terms can be used to understand the importance of various features. However, to compare the coefficients, it is important to normalize the data before training the model. The coefficients are plotted in Fig. 17. The coefficient for *ex_kurtosis_int* is much higher than the others, which makes it a deciding factor for classification. This is also evident from Fig. 4, where we can see that the distributions for the two classes are almost non-overlapping.

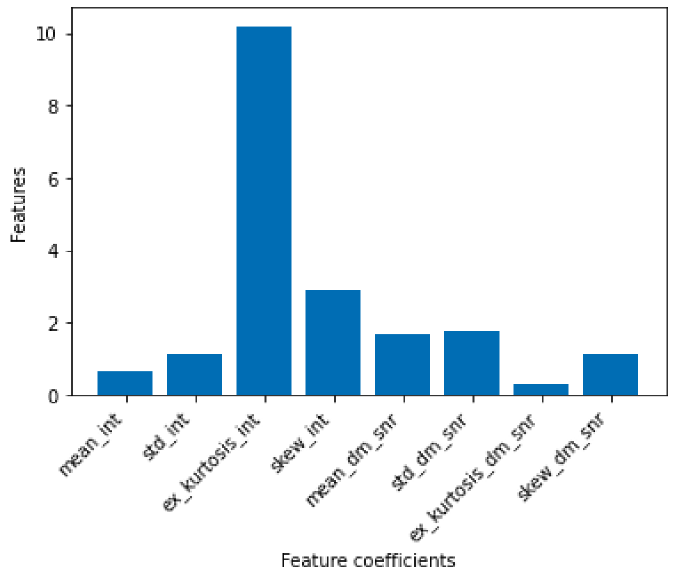


Fig. 17. Coefficients for each of the input features (normalized)

E. Using the model to make predictions

The model was used to make predictions on the test data, and the results in terms of the number of predicted pulsar stars are given in Fig. 18. Out of the 5370 data points, the model predicted **362** pulsar stars.

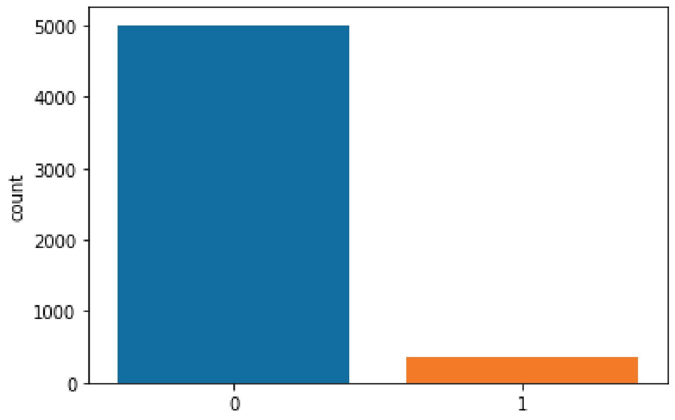


Fig. 18. Results of using the model for prediction

V. Conclusions

In this report, we analysed support vector machines and how they can be used to fit various types of classification problems by varying the kernel functions. The model was a powerful predictor that generalised well to unseen data, whose performance was measured using metrics such as accuracy, precision, recall and F1 score.

A. Avenues for Future Work

Imputation, dataset augmentation, feature selection In this work, data imputation was done using mean imputation. However, other forms of data imputation, in particular using the mode, can be tried, as the distribution is skewed for many input features. In addition, since we have identified the most important features, feature selection can be used to try and predict the target variable better.

References

[1] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani, “Support Vector Machine” in *An Introduction to Statistical Learning,* New York, Springer, 2013.

[2] Suthaharan S, “Support Vector Machine” in *Machine Learning Models and Algorithms for Big Data Classification*, Boston, Springer, 2016. https://doi.org/10.1007/978-1-4899-7641-3_9