Classification with Support Vector Machines

Vaibhav Mahapatra
Data Science Department
Indian Institute of Technology, Madras
Chennai, India
me19b197@smail.iitm.ac.in

***Abstract*—A support vector machine (SVM) is a computational algorithm that learns by example to assign labels to objects. For example, an SVM can learn to recognize fraudulent credit card activity by examining hundreds or thousands of fraudulent and non-fraudulent credit card activity reports. Alternatively, an SVM can learn to recognize handwritten digits by reviewing an extensive collection of scanned images of handwritten zeroes, ones and so forth. This article provides an overview of the simple support vector machine algorithm and how it works. The report will present the algorithm's features and how it is employed in real-life applications. This paper aims to discover the relationship between different features of a celestial object to classify it as a pulsar star.**

***Index Terms*—Classification, SVM, hinge loss, features, predictors, excess kurtosis, F1-score, margin, overfitting, classifier.**

I. Introduction

In many situations, we want our machine learning algorithm to predict several (discrete) outcomes. For example, an email client sorts mail into personal and junk mail, which has two outcomes. Another such example is a telescope that identifies whether an object in the night sky is a galaxy, star, or planet. There are usually few outcomes, and more importantly, there is usually no additional structure on these outcomes. Support Vector Machine (SVM) is a supervised learning method used predominantly for classification. The Vapnik–Chervonenkis theory is a form of computational learning theory, which attempts to explain the learning process from a statistical point of view. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier.

SVM maps training examples to points in space to maximise the distance between the two categories. New examples are then mapped into that space and predicted to belong to a class based on where they fall on the margin. SVMs can perform non-linear classification efficiently using the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces, in addition to performing linear classification. Features of the Support Vector Machine Algorithm that make it stand out in comparison to other examples are:

- It works well with a clear margin of separation.
- It is effective in high dimensional spaces with high speed computations.
- It is effective in cases where the number of dimensions is greater than the number of samples.
- It uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- SVM doesn't directly provide probability estimates, these are calculated using an expensive five-fold cross-validation. It is included in the related SVC method of Python scikit-learn library.

This paper looks at a geometric model that works on pulsar star recognition data extracted. The key focus of our work is to identify whether a star is a pulsar star or not based on the given features. We visualize and analyze the data to see if any specific elements strongly influence the target variable. Ultimately, we would like to understand the data characteristics which affect the performance of the Support Vector Classifier.

This paper is structured as follows. We first focus on establishing the necessary background and definitions. Section II focuses on the data description, analysis, handling of missing values and visualization to gain more insights into the problem. At the same time, Section III demonstrates that the use of *Support Vector Machine*. A summary and conclusions are given in Section IV.

II. Mathematics behind Support Vector Machine

The problem of binary classification is well studied, We present an approach known as the support vector machine (SVM), which solves the binary classification task. We have a supervised learning task, where we have a set of examples $x_n \in \mathbb{R}^D$ along with their corresponding (binary) labels $y_n \in \{+1, -1\}$. Given a training data set consisting of example–label pairs $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, we would like to estimate parameters of the model that will give the smallest classification error. We consider a linear model, and hide away the non-linearity in a transformation ϕ.

There are two main reasons why we chose to illustrate binary classification using SVMs. First, the SVM allows for a geometric way to think about supervised machine learning. In the papers so far, we have considered the machine learning problem in terms of probabilistic models and attacked it using maximum likelihood estimation and Bayesian inference. Here we will consider an alternative approach where we reason geometrically about the machine learning task. The SVM view of machine learning is subtly different from the maximum likelihood(ML) view. The ML view proposes a model based on a probabilistic view of the data distribution, from which an optimization problem is derived. In contrast, the SVM view starts by designing a particular function that is to be optimized during training based on geometric intuitions.

Intuitively, we imagine binary classification data, which can be separated by a hyperplane. Here, every example x_n (a vector of dimension 2) is a two-dimensional location ($x_n^{(1)}$ and $x_n^{(2)}$), and the corresponding binary label y_n is one of two different symbols. A hyperplane is an affine subspace of dimension $D-1$ (if the corresponding vector space is dimension D). The examples consist of two classes (there are two possible labels) that have features (the components of the vector representing the example) arranged in such a way as to allow us to separate/classify them by drawing a straight line.

A. *Separating Hyperplanes*

Given two examples represented as vectors x_i and x_j , one way to compute the similarity between them is using an inner product $\langle x_i; x_j \rangle$. Inner products are closely related to the angle between two vectors. The value of the inner product between two vectors depends on the length (norm) of each vector. Furthermore, inner products allow us to rigorously define geometric concepts such as orthogonality and projections.

The main idea behind many classification algorithms is to represent data in $\mathbb{R}^D$ and then partition this space, ideally in a way that examples with the same label (and no other examples) are in the same partition. In the case of binary classification, the space would be divided into two parts corresponding to the positive and negative classes, respectively. We consider a particularly convenient partition, which is to (linearly) split the space into two halves using a hyperplane. Let example $x \in \mathbf{R}^D$ be an element of the data space. Consider a function

$$f : \mathbb{R}^D \to R \tag{1}$$

$$x \mapsto f(x) := \langle w, x \rangle + b \tag{2}$$

parametrized by $w \in \mathbb{R}^D$ and $b \in \mathbb{R}$. Hyperplanes are affine subspaces. Therefore, we define the hyperplane that separates the two classes in our binary classification problem as

$$\{x \in \mathbb{R}^D : f(x) = 0\} \tag{3}$$

When presented with a test example, we classify the example as positive or negative depending on the side of the hyperplane on which it occurs. Note that (3) not only defines a hyperplane; it additionally defines a direction. In other words, it defines the positive and negative side of the hyperplane. Therefore, to classify a test example x_{test}, we calculate the value of the function $f(x_{test})$ and classify the example as $+1$ if $f(x_{test}) > 0$, -1 if $f(x_{test}) < 0$ and classify $f(x_{test}) = 0$ arbitrarily. Thinking geometrically, the positive examples lie *above* the hyperplane and the negative examples *below* the hyperplane.

When training the classifier, we want to ensure that the examples with positive labels are on the positive side of the hyperplane, i.e.,

$$\langle w, x_n \rangle + b \geqq 0 \ for \ y_n = +1 \tag{4}$$

and the examples with negative labels are on the negative side, i.e.,

$$\langle w, x_n \rangle + b < 0 \ for \ y_n = -1 \tag{5}$$

These two conditions are often presented in a single equation

$$y_n(\langle w, x_n \rangle + b) \geqq 0 \tag{6}$$

B. *Primal Support Vector Machine*

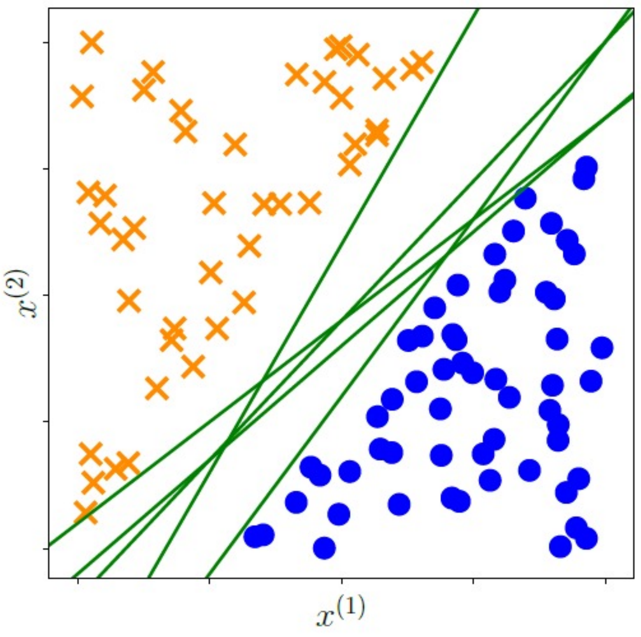

Fig. 1. Possible separating hyperplanes. There are many linear classifiers (green lines) that separate orange crosses from blue discs.

Based on the concept of distances from points to a hyperplane, we now are in a position to discuss the support vector machine. For a dataset $\{(x_1, y_1), \ldots, (x_N, y_N)\}$ that is linearly separable, we have infinitely many candidate hyperplanes (refer to Fig. 1), and therefore classifiers, that solve our classification problem without any (training) errors. To find a unique solution, one idea is to choose the separating hyperplane that maximizes the margin between the positive and negative examples. In other words, we want the positive and negative examples to be separated by a large margin (Sec. II-C). In the following, we compute the distance between an example and a hyperplane to derive the margin. The closest point on the hyperplane to a given point (example x_n) is obtained by the orthogonal projection.

C. *Concept of the Margin*

The concept of the margin is intuitively simple: It is the distance of the separating hyperplane to the closest examples in the dataset, assuming that the dataset is linearly separable. However, we need to define a scale at which to measure the distance. A potential scale is to consider the scale of the data, i.e., the raw values of x_n. There are problems with this, as we could change the units of measurement of x_n and

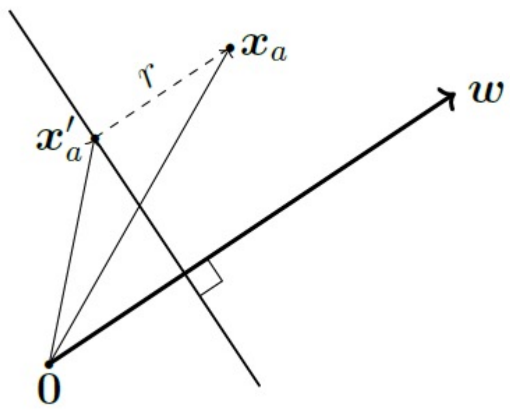

Fig. 2. Vector addition to express distance to hyperplane

change the values in x_n, and, hence, change the distance to the hyperplane. As we will see shortly, we define the scale based on the equation of the hyperplane itself.

Consider a hyperplane $\langle xi + b \rangle$, and an example x_a as illustrated in Fig. 2. Without loss of generality, we can consider the example x_a to be on the positive side of the hyperplane, i.e., $\langle hw; x_a \rangle + b > 0$. We would like to compute the distance $r > 0$ of x_a from the hyperplane. We do so by considering the orthogonal projection of x_a onto the hyperplane, which we denote by x'_a. Since w is orthogonal to the hyperplane, we know that the distance r is just a scaling of this vector w. If the length of w is known, then we can use this scaling factor r factor to work out the absolute distance between x_a and x'_a. For convenience, we choose to use a vector of unit length (its norm is 1) and obtain this by dividing w by its norm, $\frac{w}{\|w\|}$. Using vector addition, we obtain

$$x_a = x'_a + r\frac{w}{\|w\|} \tag{7}$$

we would like the positive examples to be further than r from the hyperplane, and the negative examples to be further than distance r (in the negative direction) from the hyperplane. Analogously to the combination of (4) and (5) into (6) we formulate this objective as

$$y_n(\langle w, x_n \rangle + b) >\geqq r \tag{8}$$

Since we are interested only in the direction, we add an assumption to our model that the parameter vector w is of unit length, i.e., $\|w\| = 1$, where we use the Euclidean norm $\|w\| = \sqrt{w^T w}$. This assumption also allows a more intuitive interpretation of the distance r since it is the scaling factor of a vector of length 1. Collecting the three requirements into a single constrained optimization we get our objective function as

$$\max_{w,b,r} r \text{ subject to } y_n(\langle w, x_n \rangle + b) \geqq r, \|w\| = 1, r > 0 \tag{9}$$

The (9) is equivalent to scaling the data, such that the margin is unity:

$$\min_{w,b} \frac{1}{2}\|w\|^2 \text{ subject to } y_n(\langle w, x_n \rangle + b) \geqq 1 \tag{10}$$

III. Applications in real-life problems

The Dataset we are going to analyze describes a sample of pulsar candidates discovered during the survey. A pulsar is a highly magnetized rotating neutron star that emits beams of electromagnetic radiation out of its magnetic poles. This radiation can be observed only when a beam of emission is pointing toward Earth (similar to the way a lighthouse can be seen only when the light is pointed in the direction of an observer), and is responsible for the pulsed appearance of emission. Neutron stars are very dense and have short, regular rotational periods. This produces a very precise interval between pulses that ranges from milliseconds to seconds for an individual pulsar. Pulsars are one of the candidates for the source of ultra-high-energy cosmic rays. Thus, pulsar search entails using large radio telescopes to look for periodic radio signals. Each pulsar emits a slightly different pattern, which varies somewhat with rotation. As a result, a potential signal detection known as a ”candidate” is averaged over many rotations of the pulsar, as determined by the length of observation.

A. Data Description

In the absence of additional information, each candidate could describe a real pulsar. In practice, however, almost all detections are caused by radio frequency interference (RFI) and noise, making legitimate signals challenging to detect. To facilitate rapid analysis, machine learning tools are now being used to automatically label pulsar candidates. Classification systems, in particular, that treat candidate data sets as binary classification problems are becoming increasingly popular. In this case, legitimate pulsar examples constitute a minority positive class, while spurious examples constitute the majority negative class. The data set shared here includes 16,259 spurious examples caused by RFI/noise and 1,639 genuine pulsar examples. All of these examples have been reviewed by human annotators.

Eight continuous variables and one class variable are used to describe each candidate. The first four are detailed statistics derived from the integrated pulse profile (folded profile). This is a set of continuous variables describing a longitude-resolved version of the signal that has been averaged in both time and frequency. The remaining four variables are derived in the same way from the DM-SNR curve. These are outlined below:

DM-SNR - Dispersion Measure of the Signal to Noise Ratio
Integrated profile - Folding signals w.r.t rotational period

- 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.

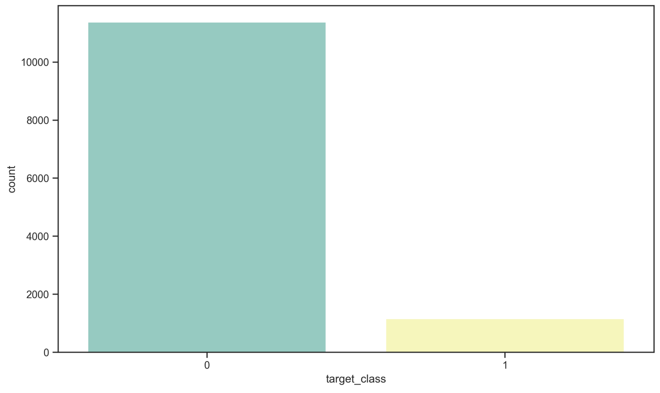

Fig. 3. target_class distribution

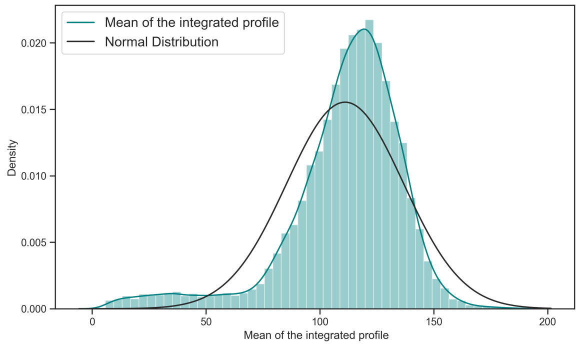

Fig. 4. Distribution of *Mean of the integrated profile*

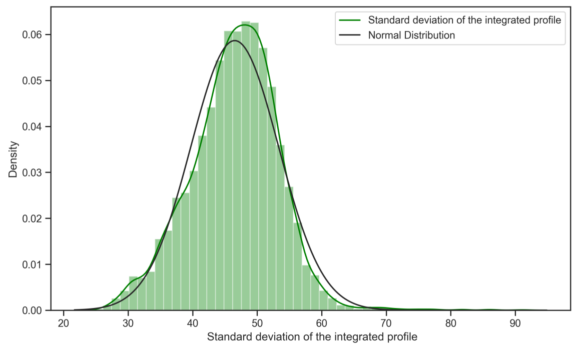

Fig. 5. Distribution of *Standard deviation of the integrated profile*

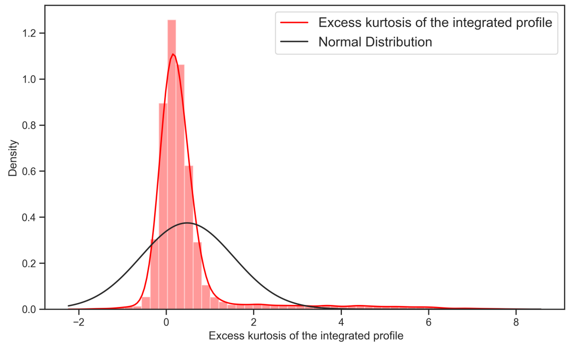

Fig. 6. Distribution of *Excess Kurtosis of the integrated profile*

- Excess kurtosis of the DM-SNR curve.
- Skewness of the DM-SNR curve.
- Class

The aim of the analysis is to determine whether is star is a pulsar star or not base using a generalised support vector machine classification model. Once the data is imported it is highlighted that there are missing values in the columns containing information about **Excess kurtosis of the integrated profile (13.85%)**, **Standard deviation of the DM-SNR curve (9.402%)** and **Skewness of the DM-SNR curve (1.8%)**. From Fig. 3 it is observed that most stars belong to target class 0. The number of missing values is significantly less. Therefore they had been imputed with the corresponding median values.

B. Exploratory Data Analysis

There are total 12528 observations in the train set and 5370 observations in the test set, with no duplicate rows anywhere. The target is NaN in test set as it is to be predicted. All the columns are numerical in nature. The data contains positive, negative, as well as null values. Some important observations we obtain pertaining to the data are regarding the mean and distribution of each variable.

The distribution of **Mean of the integrated profile** is shown in Fig. 4. We observed that the mean of the data in the column is **111.0418** with a high standard deviation of **25.672828**. The figure also tells us that the data is slightly far from a normal distribution and possesses negative skewness.

The **Standard deviation** of the Integrated profile has a near-normal distribution, as seen in Fig. 5. The mean SD is 46.6, close to the median and looking at min and max values, it seems to be generally distributed throughout the dataset.

In the **Excess Kurtosis** (Missing Values) majority of the data (atleast 75%) is less than mean. Hence a large head portion in this distribution. Hence the distribution of the left of mean is more tightly spread than the right. This means the integrated profile's tails are generally the same size as normal distributions.

The majority of this data (definitely more 75%) is less than the mean. Hence a large head portion in this distribution. Therefore the distribution of the left of mean is more tightly spread than the right. Thus the integrated profile must not be very skewed. The values are pretty high, mean being 102 and the max being 1191, which is exceptionally high compared to the 75th quartile.

For the **DM-SNR Curve** the Mean is **12.674** and the standard deviation is pretty high i.e. **29.613**, with with more

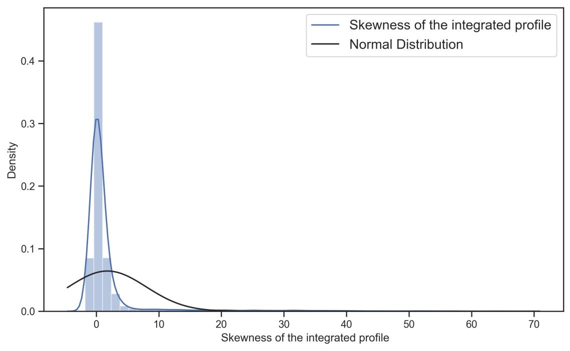

Fig. 7. Distribution of *Skewness of the integrated profile*

than 75% values being below 5.6 and the max value extremely high. We can expect the mean of most curves to be on the lower side, as seen in Fig. 8.

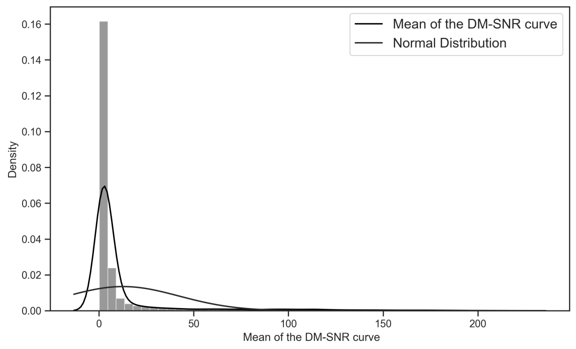

Fig. 8. Distribution of *Mean of the DM-SNR Curve*

As seen in Fig. 9 the **Standard deviation of DM-SNR Curve** also appears pretty skewed as maximum value is pretty high at 110.64 and 75% of all values are below 28, the median is 19, mean is 26, mean's higher magnitude could

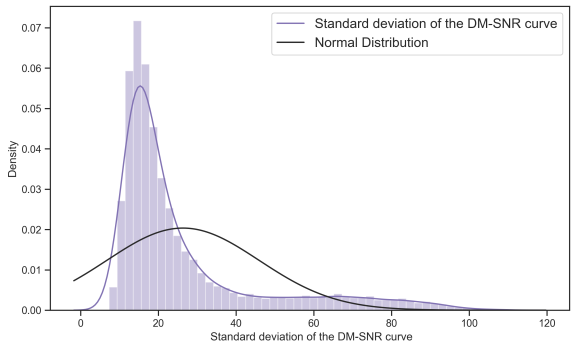

Fig. 9. Distribution of *Standard Deviation of the DM-SNR Curve*

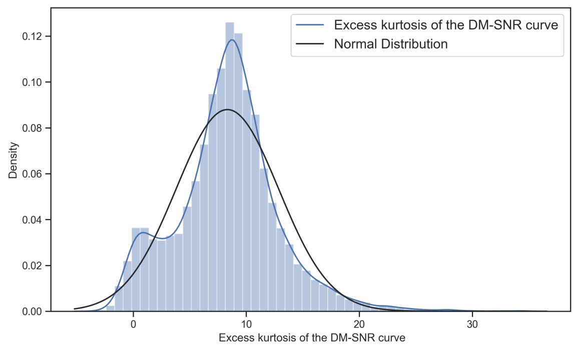

Fig. 10. Distribution of *Excess kurtosis of the DM-SNR Curve*

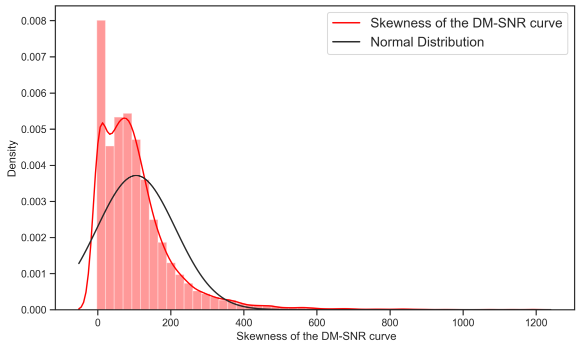

Fig. 11. Distribution of *Skewness of the DM-SNR Curve*

be a result of the extremely high values, but if they were omitted, then we could expect the standard deviation to be around the median value. The **Excess kurtosis** values are pretty high, mean being 8.23 and the max is 34, the values seem to be normally distributed with mean nearly equal to **8.32** and standard deviation of **4.53**. The distribution is illustrated in Fig. 10

Fig. 11 provides us information about the Skewness of the DM-SNR Curve. We can see that the values are pretty high, mean being **105**, the standard deviation being **107** and the max being 1191, which is exceptionally high as compared to the 75th quartile, i.e. **139**.

We find that three variables have missing values, namely *Excess kurtosis of the integrated profile, Standard deviation of the DM-SNR curve* and *Skewness of the DM-SNR curve*. For now these missing values are imputed with **median**. Further, we plotted the count of each target class, found that the data was skewed. This highly imbalanced data has 11375 non-pulsars and 1153 pulsars. The data might need sampling before the model building process. The most important metrics that we will consider to evaluate the classification models will be F1-score and Recall since the target class is significantly less.

The distribution of outliers is shown in Fig. 12. Mean of the integrated profile, Standard deviation of the integrated profile, Excess kurtosis of the DM-SNR curve and Skewness of the DM-SNR curve have outliers below 5%. Excess kurtosis of the integrated profile, Skewness of the integrated profile and Standard deviation of the DM-SNR curve have outliers between 5% and 10%. The mean of the DMSNR curve has 11.43% of the outliers. The data is visibly on different scales and requires scaling before building the model. The dataset is not normally distributed. Apart from the integrated profile's mean and standard deviation(SD), all other variables are highly right-skewed. The SD of the integrated profile appears normal but has a tail on the right side, and only the mean of IP has a heavy tail on the left side, thus left-skewed.

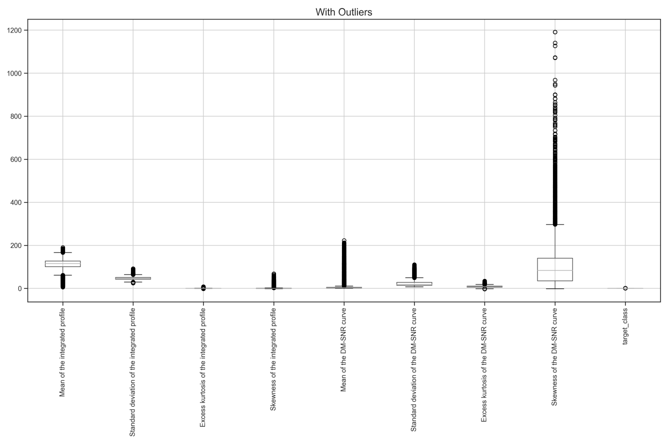

Fig. 12. Distribution of *Outliers*

After plotting a correlation heatmap (Fig. 13) between the various variables of the given dataset, we observe the following:

- **Highly positively correlated**
  1) Skewness of the integrated profile and Excess kurtosis of the DM-SNR curve
  2) Skewness of the DM-SNR curve and Excess kurtosis of the DM-SNR curve
  3) Mean the DM-SNR curve and Standard Deviation of the DM-SNR curve
- **Highly negatively correlated**
  1) Mean of the integrated profile and excess kurtosis of the integrated profile
  2) Mean of the integrated profile and skewness of the integrated profile
  3) Excess kurtosis the DM-SNR curve and Standard Deviation of the DM-SNR curve
- Excess kurtosis and skewness of the integrated profile have a high positive correlation with the target class. The mean of the integrated profile has a high negative correlation with the target class. Correlation indicates multicollinearity in data, which is undesirable during the model.

Fig. 14 describes the distribution of each data variable with respect to the **target_class**.

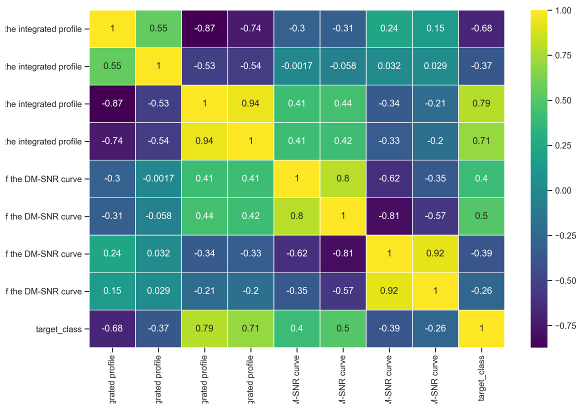

Fig. 13. Correlation Heatmap

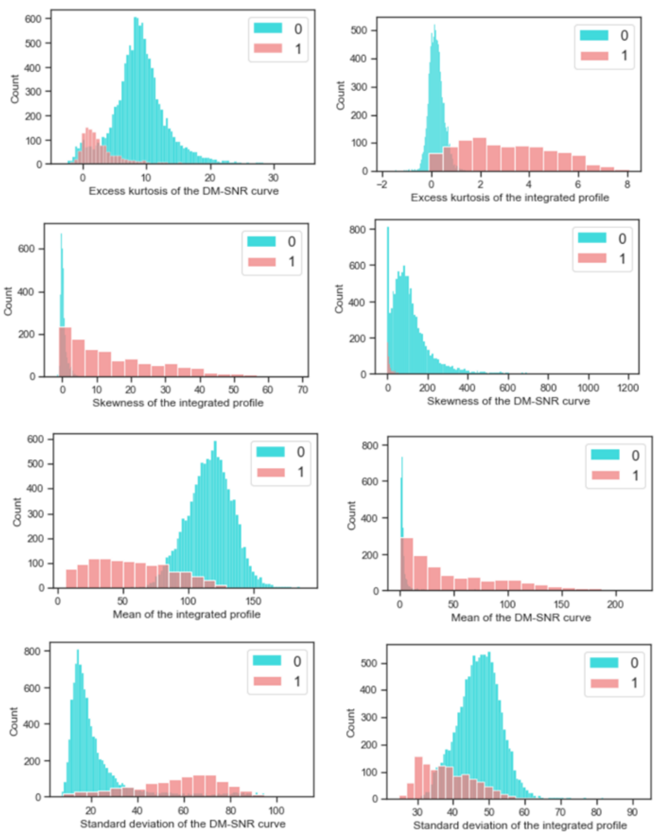

Fig. 14. Distribution of variables with respect to target_class

C. *Support Vector Machine Model and Results*

The python package *sklearn* is used for performing the task of fitting a Support Vector Machine classifier. Scikit-learn (Sklearn) is one of the most valuable and robust libraries for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modelling, including classification, regression, clustering and dimensionality reduction via a consistent interface in Python. This library, which is primarily written in Python, is built upon NumPy, SciPy and

Matplotlib. The sklearn package provides different classes for the classification problem, including **svm**. Before proceeding with the modelling, all the variables are scaled using and transformed to mean 0 and standard deviation as 1. In short, we standardize the data. It is helpful for data that has negative values. It arranges the data in a standard normal distribution. Once that is done, the data is split into train and test data. Afterwards, SVM is fitted on the training data, and the labels for test data is predicted.

The mean accuracy score on the test data using basic Support Vector Machine using Radial Basis funtion Kernal is $\mathbf{0.9757}$ where as the test accuracy is $\mathbf{0.90774}$, which indicates that our model is performing decently. The majority of stars lies in the category $\mathbf{0}$. The **null accuracy** obtained is $\mathbf{0.9084}$, which therefore doesn't convey information about whether the classifier is doing a decent job or predicting the majority class only. The 10 fold cross-validation score for the training data is $\mathbf{0.9741}$ and for the test data it is $\mathbf{0.9753}$. The confusion matrix describes the following:

$$\begin{bmatrix} 3389 & 26 \\ 65 & 279 \end{bmatrix}$$

- True Positives(TP) = 3389
- True Negatives(TN) = 279
- False Positives(FP) = 26
- False Negatives(FN) = 65

The train and test classification report suggest the following

	precision	**recall**	**f1-score**	**support**
0	0.98	0.99	0.99	7960
1	0.93	0.80	0.86	809
accuracy			0.98	8769

TABLE I
CLASSIFICATION REPORT OF THE TRAINING DATA

	precision	**recall**	**f1-score**	**support**
0	0.98	0.99	0.99	3415
1	0.91	0.81	0.86	344
accuracy			0.98	3759

TABLE II
CLASSIFICATION REPORT OF THE TEST DATA

Since the results we obtained are satisfactory enough so we trained the data as whole and predicted the value on the given test set after performing scaling and removing the outliers.

To find out the best hyper-parameters for SVC model, we employ scikit-learn's RandomSearchCV method to do a grid search with the below given parameters:

- 'C': [0.01,0.1, 1, 10, 100, 1000],
- 'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
- 'kernel': ['rbf'],
- 'tol':[0.01,0.001,0.0001],
- 'degree': [2,3,4,5]

The grid search yielded the following results for the best model - **SVC(C=100, degree=5, gamma=0.001, random state=1)**. The confusion matrix for the best model looks as follows:

$$\begin{bmatrix} 3397 & 18 \\ 57 & 287 \end{bmatrix}$$

The classification report suggest the following:

	precision	**recall**	**f1-score**	**support**
0	0.98	0.99	0.99	3415
1	0.94	0.83	0.88	344
accuracy			0.98	3759

IV. CONCLUSION

To date, only over 2,000 pulsars have been detected. The task of *classifying whether a star is a pulsar or not* is beneficial for scientists as well as astronomers for their celestial research. The given data is used above to describe the relationship between different variables and identify the most useful ones. Before modelling, we dropped the variables like Excess kurtosis of the integrated profile, Skewness of the DM-SNR curve and Standard deviation of the DM-SNR curve due to their high correlation with other variables. The EDA and Support Vector classifier approach gives a good idea about the relationship between predictors and regressors. It explains the underlying pattern thoroughly and predicts well.

From our observations, we can conclude the following about the significant features:

- **Mean of the integrated profile**: The mean of the integrated profile is lesser for pulsar stars than nonpulsars.
- **Excess kurtosis and skewness of the integrated profile**: For pulsars, the excess kurtosis and skewness are higher and more spread out as compared to non-pulsars.
- **Mean of the DM-SNR curve**: The mean of the integrated profile is higher and spread out for pulsar stars than non-pulsars.
- **Excess kurtosis and skewness of the DM-SNR**: For pulsars, the excess kurtosis and skewness are lower and less spread out as compared to non-pulsars. Thus, as shown by both the qualitative and the quantitative analysis, the pulsar stars can be classified depending on their various features.

REFERENCES

[1] https://drive.google.com/file/d/13aIxXP5BgX9adIFuX8Xigg-3VZPUtX8s/view

[2] Deisenroth, Marc Peter, Faisal, A. Aldo and Ong, Cheng Soon. Mathematics for Machine Learning. : Cambridge University Press, 2020, pp.370–380

[3] Boyd, Stephen, and Vandenberghe, Lieven, Introduction to Applied Linear Algebra, Cambridge University Press, 2018

[4] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani. An Introduction to Statistical Learning : with Applications in R. New York :Springer, 2013, pp.344–355

[5] https://stackabuse.com/implementing-svm-and-kernel-svm-with-pythons-scikit-learn/