



# Visual Explanation Of Machine Learning (SVMs)

**Final Year Dissertation**

Pranav Pai

BSc Computer Science (Hons)

Supervisor: Prof. Mike Chantler

Second Reader: Prof. Marko Doco

## DECLARATION

I, Pranav Pai confirm that this work submitted for assessment is my own and is expressed in my own words. Any uses made within it of the works of other authors in any form (e.g., ideas, equations, figures, text, tables, programs) are properly acknowledged at any point of their use. A list of the references employed is included.

Signed: *Pranav Pai*

Date: 21st April 2022

## Abstract

One of the most widely used Machine Learning Classifiers is the Support Vector Machine (SVM). It belongs to the field of Supervised learning algorithms that classifies between classes using the concept of margin. It outperforms KNN, Decision Trees, and Naive Bayes Classifiers in terms of accuracy, and is thus quite effective. They offer two major advantages over modern algorithms like neural networks: faster processing and better performance with fewer samples. Within these supervised learning algorithms there are different types of SVMs, in particular they are the examples of the “kernel trick”. These are the polynomial kernel, gaussian radial basis function, sigmoid kernel and more.

The working of these machine learning techniques could be very well understood by visualisations, mainly the ones where the users could interact with the interface in many ways. These then help in understanding how the algorithm finds the best fit model by using the best sets of parameters. There are very few interactive models that help give users in depth knowledge regarding SVMs and how they work. With respect to the current visualisations, this technical research project dives deep into the different types of parameters with various types of kernels that could affect the model by presenting an interactive visualisation. In-depth semi-structured interviews with six participants (knowledgeable about machine learning) are used to assess this visualisation with 3 new users and 3 experienced users. What we find after the interview sessions is that interactive visualisation has successfully proven to help the users gain an understanding about the parameters that helps classify the data with support vector machines.

## Terminology

The SVM domain contains a big database of technical terms. To avoid misunderstandings, the terms and definitions listed below will be used throughout this work.

Term	Definition
<b><i>SVMs</i></b>	Support Vector Machines are supervised machine learning models that help in classification and regression analysis.
<b><i>ML</i></b>	Machine Learning is a type of AI that makes a machine have human-like capabilities. Especially while making decisions on given data.
<b><i>Kernels</i></b>	A method to classify non-linear data using a linear classifier method.
<b><i>Vectors</i></b>	Vector is a point in space which encodes a length and direction. They are usually 1-dimensional. They help in organising data in machine learning
<b><i>Hyperparameters</i></b>	They are certain parameters that are used to control the learning algorithm. They are usually pre-defined before the working of the algorithm.
<b><i>Feature</i></b>	A feature is an independent measurable property or characteristics.

## Contents

<b>1. Introduction</b>	<b>7</b>
1.1 Motivation	8
1.2 Aim and Objective	8
<b>2. Literature Review</b>	<b>10</b>
2.1 Reflection on Literature from First Deliverable	10
2.2 Background	11
2.2.1 Core Components and Theory of SVMs	12
2.2.4 Summary	28
2.3 Critical Analysis of existing visualisations	29
2.4 Conclusion and Future work	32
3.1 Functional Requirements	34
3.2 Non-Functional Requirement	34

<b>4. Design</b>	<b>35</b>
4.1 Languages and Libraries	35
4.2 User Interface Design	37
4.3 Summary	41
<b>5. Implementation</b>	<b>42</b>
5.1 Additional Resources	42
5.2 Development	42
5.2.1 Back-end	43
5.2.2 Front-end	48
5.3 Testing	50
5.3.2 System Testing	52
5.4 Summary	52
5.4.1 Critical reflection and future changes	53
<b>6. Evaluation</b>	<b>54</b>
6.1 Methodology	54
6.1.1 Participant recruitment	55
6.1.2 Ethical and legal considerations	57
6.1.3 Procedure	57
Task 1 - analysing visualisations	59
Task 2 - observational use	59
6.1.4 Pilot test	60
6.2 Results	60
6.2.1 Qualitative results	60
6.2.2 Quantitative results	61
6.3 Findings	63
6.4 Summary	65
6.5 Personal Reflection on Evaluation Design	66
<b>7. Project Conclusion and Future Work</b>	<b>67</b>
<b>8. References</b>	<b>68</b>
<b>9. Bibliography</b>	<b>70</b>
<b>10. Appendix</b>	<b>71</b>
10.1 APPENDIX A - The dashboard of the application (UI)	71
10.2 APPENDIX B - Snippet of Different ways to insert data (UI)	71
10.3 APPENDIX C - User Study Consent Form	72
10.4 APPENDIX D - NASA Task Load Index (TLX) Questionnaire	73
10.5 APPENDIX E - Interview Script	74
10.6 APPENDIX F - Gantt Chart	75
10.7 APPENDIX G - Declaration Form from University	76

## List of Figures

1. Linear support vector .....	11
2. Optimal hyperplane in classification.....	12
3. best hyperplane .....	14
4. identifying best hyperplane in SVM .....	14
5. Support vectors.....	15
6. Separation using hard margins .....	16
7. Best hyperplane for multidimensional data .....	17
8. Linear kernel.....	19
9. RBF kernel.....	19
10. Polynomial kernel.....	20
11. Sigmoid kernel.....	20
12. Effect of C parameter .....	22
13. Effect of gamma parameter .....	23
14. Existing interactive tool for SVM.....	29
15. Showing RBF function on the visual tool.....	31
16. Reactive graph design of application.....	37
17. Test size slider.....	38
18. kernel selection drop down menu .....	38
19. threshold frequency knob.....	39
20. Cost 'C' slider.....	39
21. Gamma slider.....	39
22. Noise level slider.....	40
23. ROC curve graph.....	41
24. Confusion Matrix.....	41
25. Non interactive visual tool .....	58
26. Dashboard of the application programmed .....	58
27. NASA TLX Results .....	62

## 1. Introduction

The SVM algorithm was originated by Alexey Ya. Chervonenkis and Vladimir N. Vapnik, 1963. In 1992, Isabelle Guyon, Vladimir Vapnik, and Bernhard Boser suggested a new way to create non-linear classifiers by two methods. They applied the kernel trick to the maximum-margin hyperplanes.

An SVM or support vector machine is a widely used learning method in text analysis, bioinformatics, and computer vision. An SVM model is a model that tries to isolate given training data in two classes with a hyper-plane at the place where two classes are ideally far from each other. The class of another data point is anticipated by figuring out which side of the hyper-plane it lies on. This is one of the simplest definitions of support vector machines. To most SVMs still remain a black-box model in the world of AI and machine learning. Other machine learning algorithms such as decision trees and linear regression are fairly easier to understand on a beginner basis. However SVMs are complex models and get even more complex when there are models which represent classification in higher dimensions. This makes learning more difficult for new users learning the technology for the first time, and it can be a barrier for professionals wanting to enhance their models. For years, model performance was typically observed by numerical statistics; however, detailed visualisation methods have subsequently grown popular to overcome these challenges. These interactive visualisations provide users with great insights of model behaviour and introduce new levels of understandability that can support young learners and experts alike in learning, diagnostic, and development tasks.

This project captures the whole journey of researching current and improved visualisations for SVMs. We will then dive deep into the design and implementation of an improvised version of an existing visualisation tool (created using python) which will help users understand the SVM model (with different kernels) and the parameters that affect the model in many ways. This tool will be thoroughly evaluated in terms of its performance, which will assist us in answering the research questions listed

below. In the motivation section, this document also details important topics of personal reflection during the project.

## 1.1 Motivation

There has been very little research and experiments done to make students and professionals understand the actual mathematical meaning behind these ML algorithms. Most just simple copy and paste code without knowing the mechanics behind each algorithm. Therefore improvement in this field for the users will remain stagnant and there would be no room to grow further if their basics aren't clear. Hence better visualisation tools are a must to enhance the explainability of these machine learning models to new and experienced users.

Humans are very visual in nature. And add interactivity with a bit of fun to that, they become more engaged and susceptible in learning new things. On a personal level, I have been very fascinated by machine learning and data science. This project will definitely strengthen my research skills as well as my knowledge in this field. This project will help me in my advancement towards pursuing masters.

## 1.2 Aim and Objective

The aim of this project is to discuss and evaluate the improved visualisations of SVMs. The literature review of this project focuses on these two important research questions regarding SVMs.

They are :-

1. Does the interactivity of hyperparameters ('C', 'gamma') improve the model performance and reduce cognitive load on the subjects' minds over existing visualisation tools?
2. Does the SVM visualisation tool containing any sort of interactivity such as, customizable data distribution, kernel types and other hyper-parameters provide improved explainability for all types of users?



The following objectives will be used to answer these questions:-

1. By performing a deep research analysis on the literature of the topic and trying to understand the gaps that need to be looked upon.
2. Mentioning the current visualisation tools available and pointing out its weaknesses and studying them further to help us make a better improved version.
3. Identifying the key requirements needed for our interactive visualisation.
4. Explaining the design and implementation process of the application.
5. Conducting an in-depth qualitative user interview and survey to gain valuable insight into the application.
6. Giving an overview of the findings and talking about the future development of these types of visualisation tools and ways it could impact user explainability.

## 2. Literature Review

The purpose of this chapter is to examine and critically analyse the important areas of literature relevant to the project, allowing gaps in the literature to be found and project requirements to be developed. The work is divided into three sections; the first discusses on the research that was done in the first deliverable and highlights the key points; the second is about the background which conceptualises SVMS with respect to the current research giving an in-depth analysis about the components of SVMs and the challenges faced; and the third is a critical analysis on the existing visualisation tools available and how better visual tools could be built to overcome these challenges.

At the end we discuss the future work that could be done with the knowledge that has been gained with this chapter. It is recommended that the readers of this project already have the basic knowledge on machine learning, SVMs in particular.

### 2.1 Reflection on Literature from First Deliverable

During the first deliverable, the main focus was on making users understand the key aspects of SVMs and how they function in theory. Literature on support vectors and their use in classification and regression models were covered. One of the main topics, the “kernel trick” was explained. The basic calculations carried out between two classes of data sets were explained for better understanding of how the SVMs grouped their training data. In and all the deliverable focused more on visualisations of SVMs in terms of statistical graphs which helped in explainability to users.

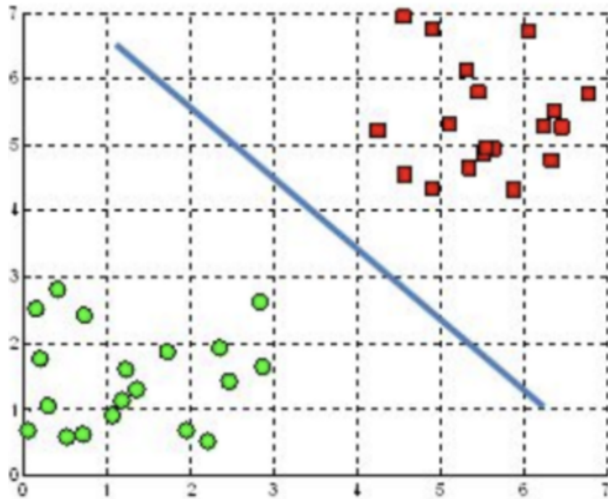


Figure 1: linear support vector used as a classifier (Gandhi, R., 2018)

Since then the project has been thoroughly researched on improving the existing visualisation tools in order to widen the user base. And this literature review will go through different aspects in creating great interactive visualisations for the explainability of SVMs and its parameters that affect the model.

## 2.2 Background

To support in understanding the rest of this document, it is mandatory to conceptualise and discuss the core components of SVMs. This chapter would help new learners identify the challenging aspects of understanding SVMs, and this knowledge combined with the existing studies would help in better explainability of this so-called black-box algorithm. We will initially build-up with the basic definitions and then recognize these existing technologies before trying to fill up gaps and come to a concluded solution with our application.

Thus, the literature should cover the topics as followed :-

- Describe the core components of SVMs.

- Discuss about different parameters that affect the classification of data and how they help in finding the best fit model in theory.
- Describe which part of SVMs the users have difficulty understanding.
- Describe how the difficulties could be solved with improved visualisations.

### 2.2.1 Core Components and Theory of SVMs

In this topic we will go through research papers and articles on different core components of SVMs such as different types of kernels, parameters that affect the model and also the functioning of SVMs. Their work will also cover the architecture of the model and show how each vector component is calculated in the dataset. This will help understand the function layer by layer. This work is divided into major key components as follows.

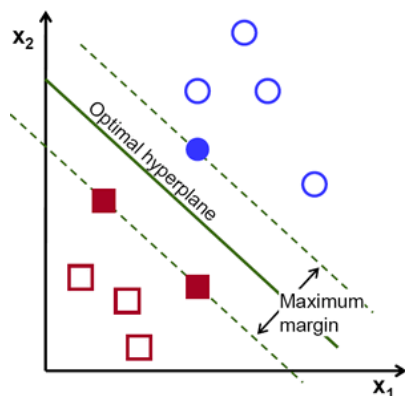


Figure 2: optimal hyperplane in classification model of SVM(Gandhi, R., 2018)

#### **What are Support Vector Machines and how do they work ?**

The Support Vector Machine, or SVM, is a linear model for solving classification and regression problems. It is suited for a wide range of applications and can tackle both linear and nonlinear problems. SVM is a fundamental concept: The approach uses a barrier or hyperplane to partition the data into classes (Rushikesh,2018). Every data point is plotted as a point in n-dimensional space

(where  $n$  is the number of features you have), with the value of each feature becoming the value of a specific coordinate in the SVM algorithm (Sunil,2017). SVM categorises data points by mapping them to a high-dimensional feature space, even when the data is not otherwise linearly separable. After finding a partition between the categories, the data are converted so that the separator can be drawn as a hyperplane. Following that, provided data features can be utilised to determine which category a new data should belong to (IBM,2017). Because the model's complexity is  $O(n\text{-features} * n^2 \text{ samples})$ , it's ideal for working with data in which the selection of attributes exceeds the number of samples.

### ***Hyperplanes***

Hyperplanes are decision boundaries that help categorise data. Data points along either side of the hyperplane can be allocated to different classifications. The hyperplane's dimension is also determined by the number of features. If there are only two input characteristics, the hyperplane is solely a line. When the number of input characteristics approaches three, the hyperplane becomes a two-dimensional plane. It is impossible to imagine when the number of attributes exceeds three (Gandhi,2018). It's the one that optimises the margins from both tags, as per SVM. To put it another way, the hyperplane (in this case, a line) that has the largest distance to the feature's closest element.

In order to create a best fitting model on a linear dataset, the model needs to divide the dataset the best way possible. For that to happen, the model needs to have the best hyperplane. The best hyperplane is the one that optimises the margins in both tags, per the SVM model (See Fig 1.4). In other words, the hyperplane (in this case, a line) with the longest distance to the nearest element of each tag.

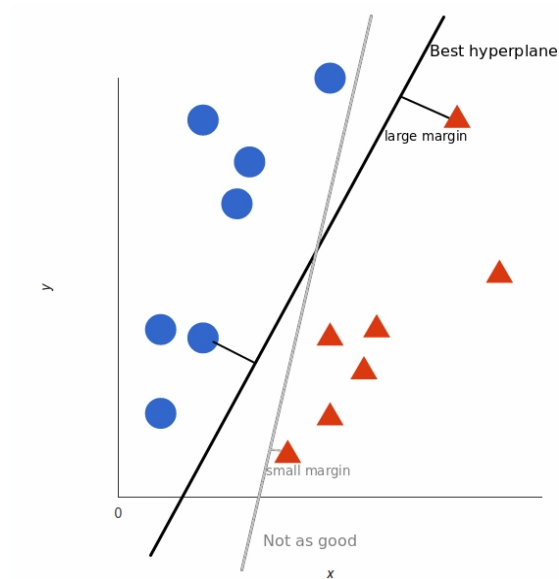


Figure 3: Best Hyperplane (Sunil,2017)

The mathematical formula to calculate the hyperplane in linear equations.

$$a_0 + a_1x_1 + a_2x_2 + \dots + a_nx_n$$

The hyperplane's intercept is denoted by  $a_0$ . The first and second axes are also defined by the letters  $a_1$  and  $a_2$ . Two dimensions are represented by the letters  $x_1$  and  $x_2$ .

The two main concepts that help to mark the best line is **Maximum Classification** and **Best Separation**. Let's take an example to get to know how exactly this works (Shipra, 2017).

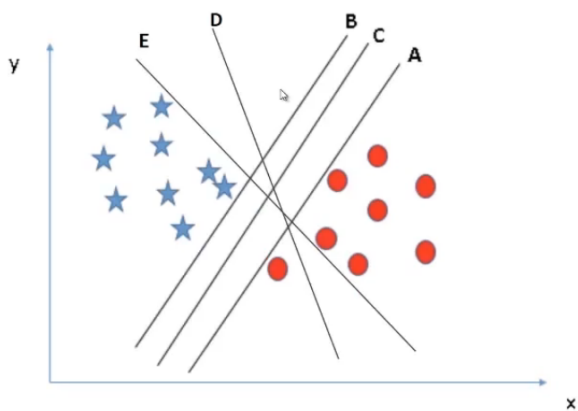


Figure.4: Identifying the best hyperplane (Shipra,2017)

Here, in the above picture there are five decision boundaries or hyperplanes that are separating the dataset. Here, by using the two rules stated above, the hyperplane E and D can be scrapped as the best hyperplane. Since these lines are misclassifying the red data-point, since we need a maximum amount of correctly classified data-points it cannot be considered as the best. When it comes down to B,C or A hyperplanes. C is the best out of those three by following the rule of Best Separation. The hyperplane A is too close to the red data-points, there is more space on the blue side than the red. This means that the model with that hyperplane will have a very small chance to misclassify the blue data-point, but will easily misclassify the red one. Similarly, the model with hyperplane B will easily misclassify the blue data-point. Hence both these hyperplanes are not considered the best hyperplane.

### **Support Vectors**

The data points closest to the hyper-plane are called support vectors, and they influence the hyper-position planes and orientation. We must choose a hyperplane with the maximum margin, i.e. the length between the support vectors and the hyperplane. The hyper-plane can be changed by even minor interference in the position of these support vectors (Riya Chourasiya,2021).

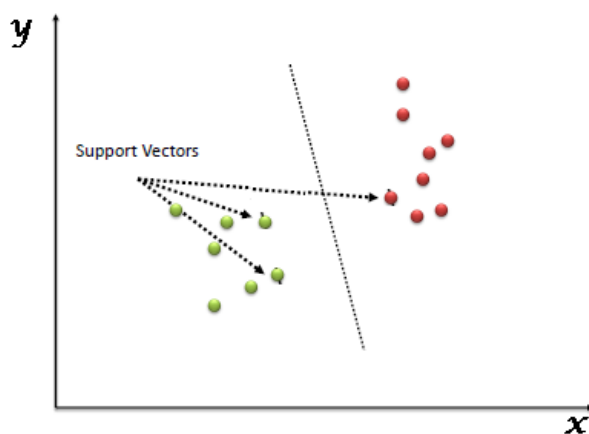


Figure 5: Support Vectors(Sunil,2017)

## Margins In SVMS

There are two types of margins used to define certain aspects of SVMs. The **Hard Margin** and the **Soft Margin**. The margins are the space between the hyperplane and the support vector. We usually use hard margins when the data is linearly separable and there is no misclassification that is required.

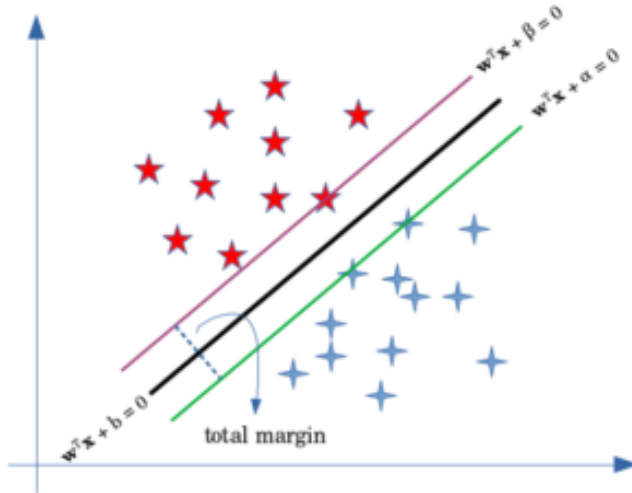


Figure 6: Separation of data using Hard Margins

Here  $w^T x + \alpha = 0$ , and  $w^T x + \beta = 0$  define the two parallel lines. In the figure there is no such misclassification of the data. And so the optimal solution for the best fit model is to maximise the margins. The distance formula for both the red data-points and the blue data-point from the black line are as follows;

$$\frac{|w^T x + \alpha|}{||w||} \quad \text{and} \quad \frac{|w^T x + \beta|}{||w||}$$

The result for the total margin would be,

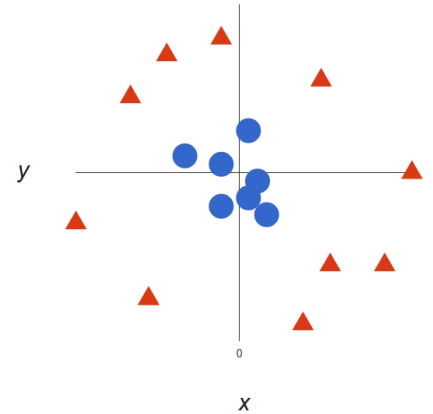
$$\frac{|\alpha - \beta|}{||w||}$$

Conversely, soft margins are used in SVMs when the data is not linearly separable and hence there would require a small amount of misclassification to avoid overfitting the model (Karimi,2021).



### ***What happens when the data cannot be separated linearly?***

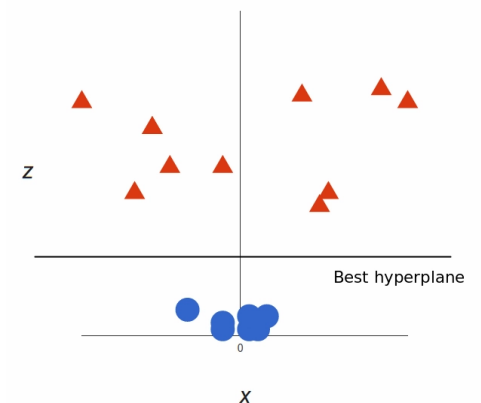
Not all datasets can be classified by a straight line, like the one in figure. Here it can be noticed that the data are in somewhat a circular form. This is called Non-linear data. The simple solution to classifying this dataset is to project them in higher dimensions. Up till now the dimensions that were discussed were restricted to 2-D. However these data can be projected into a third dimension, here as seen in the figure. the vectors could be easily segregated.



Lets say the dimensions in 2-D were X and Y. Now dimension Z has been introduced. By using the formula ,

$$z = x^2 + y^2;$$

The formula used above is that of a circle. So when the data is projected into a higher dimension the graph would look like the one in the figure. Now at this stage it is very easy to find the best hyperplane. This process of mapping the dimension Z into the already existing dimension X and Y is known as **Kernel Transformation**. The kernel, to put it differently, accepts the features as input and generates linearly separable data in a multidimensional space.



After the kernel transformation, the data is driven back into its normal state

of two dimensions and the hyperplane has been mapped accordingly. As shown in figure 6.

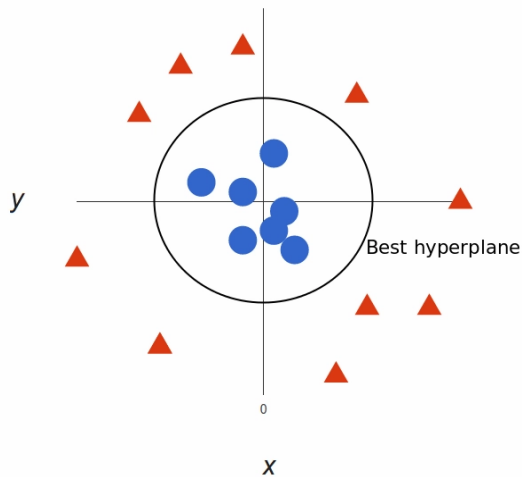


Figure 6: Best hyperplane for the multidimensional data

Non linear SVM means that the data that is captured is not linearly separable, but the benefit is that more complex structural dataset can be classified using this algorithm and formulated relationships within the data could be found. However the disadvantage is that this takes a long time and requires a lot of computing power (Yhat blog).

### **The Kernel Trick**

The kernel transformation discussed above cannot be manually done by us. It is computed by the machine learning algorithm SVM itself. Here the “trick” is that the SVM algorithm doesn’t need the actual vectors to work. It simply needs the dot product of the vectors.

$$\mathbf{a} \cdot \mathbf{b} = x_a \cdot x_b + y_a \cdot y_b + z_a \cdot z_b$$

$$\mathbf{a} \cdot \mathbf{b} = x_a \cdot x_b + y_a \cdot y_b + (x_a^2 + y_a^2) \cdot (x_b^2 + y_b^2)$$

Here “a” and “b” are the vector points in the dataset and the x,y,z are the coordinates on the plane.

This is the kernel technique, which saves energy / cost by avoiding multiple expensive calculations. We get a linear classifier when the kernel is linear. We can get a nonlinear classifier without changing the data, by using a nonlinear kernel there is a sudden adjustment of the dot product to the desired space, and the SVM algorithm just plays its role after the “kernel trick” . By using this trick, the decision boundary of a linear or non-linear dataset could be drawn. The kernel trick is not a part of SVM. It is used in other classifier methods as well, such as logistic regression models (Bruno,2017). There are many types of kernel functions, each is used depending on the complexity of mapping the data.

### ***Different Types of Non-linear Kernels***

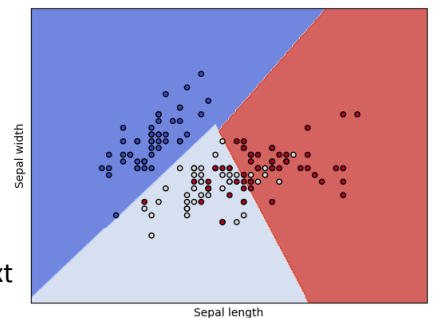
This subsection goes through different types of Kernels that help SVMs during classification.

#### **1. Linear Kernel**

The Linear kernel function work best when the dataset is linearly separable. It is one of the most commonly used kernel functions.

It is also mainly used when there are a large number of features

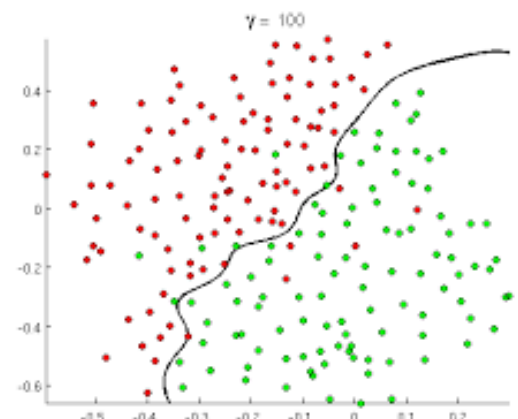
In the dataset (as shown in figure). Linear Kernels are used for Text



Classification, since each alphabet is a different feature with different properties. The main advantage of using this function is that the training time is significantly less than most kernel functions and there is only one tuning parameter that affects this function, the C regularisation parameter which will be discussed later on (GeeksForGeeks, 2018).

#### **2. Radial basis function (RBF) kernel**

The RBF function is popularly used to classify non linear dataset. As shown the the figure, the function uses soft margins over the dataset to



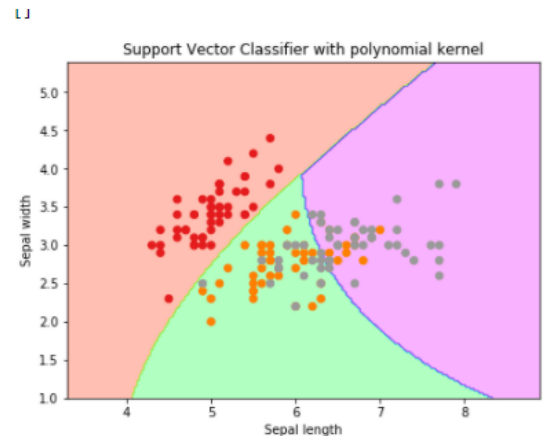
avoid underfitting the model. However they are sometimes prone to overfitting. This can be controlled by the use of tuning parameters. Overall this function is a powerful function used in SVMs (Andre,2020).

### 3. Polynomial kernel

The polynomial kernel is also a function used mainly on non-linear datasets. This function,

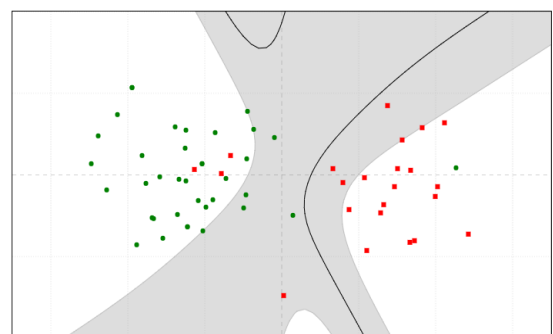
$$K(X_1, X_2) = (a + X_1^T X_2)^b$$

where  $a$  = constant term and  $b$  = degree of kernel, is used to calculate the dot product of the vectors. The degree of the polynomial increases as the dimensions increase, hence it is computationally more expensive than the RBF function. However, this function is more accurate and gives the best combination of the base features of the dataset during classification (Shipra,2021).



### 4. Sigmoid kernel

Sigmoid function also known as the hyperbolic tangent kernel comes from the neural network background. Since it is here used in



A 2-D perspective, it can be described as an equivalent function to a two-layered perceptron. Where the function of a hyperbola is used to compute the dot product of the vectors. Studies show that the sigmoid function works well in practice and has a good number of

practical real use cases, but less accurate than the RBF function (Hsuan-Tien Lin and Chih-Jen Lin, 2018).

### ***Parameters that affect SVMs***

Finding the right kernel function for the dataset to classify the features is most important. However tuning the SVM model with hyperparameters that have an effect on the backend of these functions is equally important. These hyperparameters are effective in improving the model performance. They are not directly used in the sci-kit learning library but passed as an argument in the estimator function. The research questions associated with this project are based on the two main hyperparameters **C Regularisation** and **Gamma Parameter**.

#### **'C' Parameter**

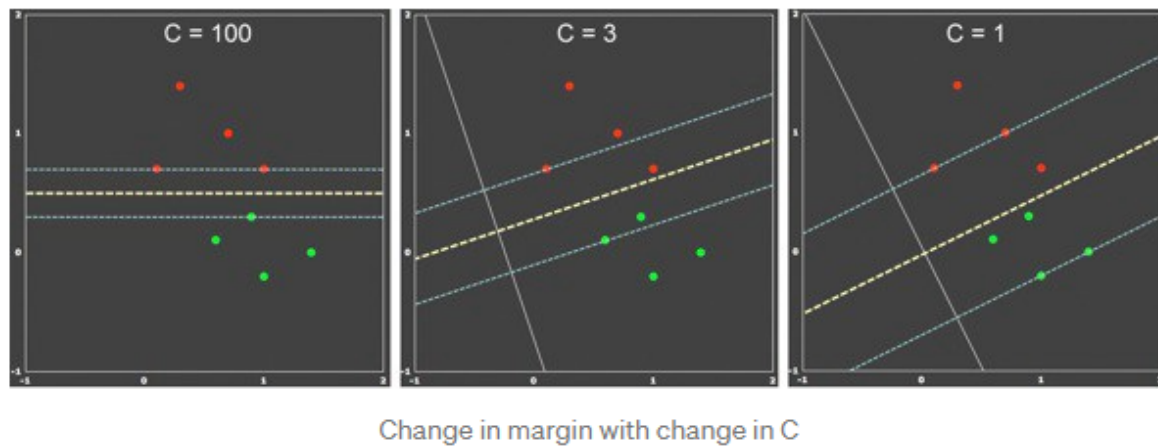
The C parameter is a hyperparameter for error control. It tells the SVM optimization how much you don't want each training example to be misclassified. For large values of C, the optimization will choose a relatively small hyperplane if it does a better job of accurately classifying all of the training points. A small value of C, on the other hand, will drive the optimizer to look for a hyperplane with a greater margin separating it, even if that hyperplane misclassified more points. The effect of incrementing the hyperparameter C is to make the margin tighter and, thus, less Support Vectors are needed to define the hyperplane. And the effect of decreasing the parameter means that the model won't be underfitting and will be more generalised, therefore classifying outliers correctly. Even in cases where the training data is linearly separable, the SVM will misclassify samples for very small values of C (Carlos,2021).

**Large Value of parameter C => small margin => Smaller Error**

**Small Value of parameter C => Large margin => Large Error**

The C values are usually the values like 0.001, 0.01, 0.1, 1, 10, 100.

The figure below will specify how exactly the tuning of SVMs work;



*Figure 7: Showing the effect of parameter C in classification of data*

When it comes to bias and variance,

- Low bias and high variance are the results of a large C. Low bias since the cost of misclassification is heavily penalised.
- With a small C, you get more bias and less variance.

The C parameter can be tuned in all the kernel functions. Note that the hyperparameter can give different forms of results when used with different types of kernels. Overall it plays an important role in the significance of finding the best fit model for a given dataset during classification.

## 'Gamma' Parameter

The Gamma Parameter, also denoted as  $\gamma$ . This parameter is only used in non-linear classification. It is mainly used in RBF function before training the model. It adds weight to the curvature of the decision boundary (hyperplane). The gamma parameter determines how far a single training example's influence extends, with low values indicating 'far' and large values indicating 'near.' In other words, when the gamma is low, points far away from the plausible separation line are taken into account in the separation line calculation. High gamma, on the other hand, suggests that the points closest to the plausible line are taken into account while calculating (Carlos,2020).

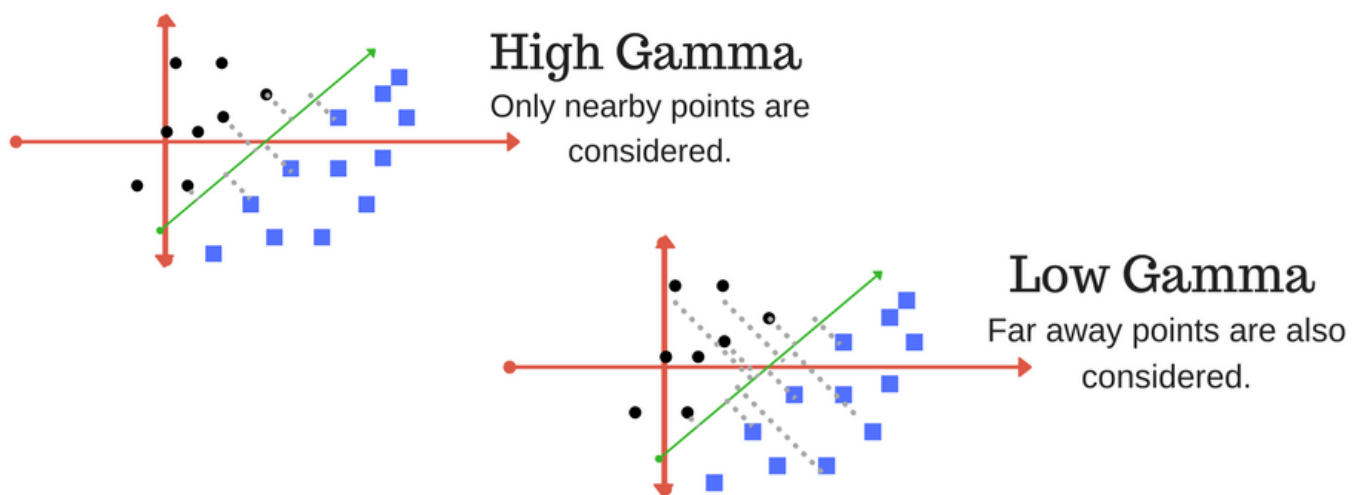


Figure 8 Showing the effect of gamma parameter in classification of data

**Gamma high** means **more curvature**.

**Gamma low** means **less curvature**.

**To summarise**, we have covered the main theory and the components of SVMs that will be useful going ahead with the project. Next section talks about challenges faced in understanding SVMs.

### 2.2.2 Challenging Aspects in understanding SVMs

Despite the generalisation ability and some advantages of SVM, there are some challenges faced by users whether they are new or experienced during studying and implementing SVM. There are many important aspects of SVMs that are quite difficult to understand while implementing the algorithm on their dataset, mainly due to their complex structure and functions that operate these algorithms. This subsection will steer the literature to give a base to our project requirements, that will help overcome these challenges. These challenges are with respect to both new and experienced users.

Further research into this topic, shows that SVMs are usually understood by referencing to online materials available, such as images, videos and graphs. There isn't much explanation on how these work, especially on a visual basis. New users tend to try to understand these algorithms by straight diving into the mathematical functions. This imposes a lot of frustration to the new user as it may be their first time learning about this algorithm. A survey done found that most users are not familiar with algebraic structure and physics that is needed to understand the basic dot product calculation that forms the basis of the classification in SVMs. It is very important to understand the basic concept of how SVMs function on a step by step process. So that the user is capable of handling different sorts of data and finding the best fit model in terms of classification. Here new users are not able to get through the initial stage of understanding the algorithm, hence it is difficult for them to grasp further topics such as kernels and hyperparameters.

As for the experienced users, they might be able to have a solid understanding of how the SVMs work and have the knowledge of their structure, but there are also many instances that ML developers have hit a certain roadblock while implementing these algorithms due to lack of control on the model from the developer's end. This is also because of the lack of explainability, where the model is referred to as the 'black box' model. Sometimes there can be very few tools available for the



developers to play with their model hence a huge hindrance to the performance in classification. Therefore it is key to know the reasoning behind these structures and also what are the major parametric tools that will help improve performance on these models. Here are some key highlights on the challenges faced by the majority of developers who use SVM for classification.

- **Algorithmic Complexity:** Due to excessive computational cost and due to the growth of training kernel matrix in quadratic form, while implementing the visualisation, users faced the challenges related to complexity. This complexity is because SVMs are not suitable for large dataset classification. To reduce the complexity or the size of training sets, optimal robust selection scheme is required.

- **Optimal classifier development for multi-class problems:** Support Vector Machines were created to handle binary classification difficulties in the first place. The multi-classification problem for SVM does not have an easy solution (Dr. .M.Mohamed Sathik,2010). To use SVM to multi-classification problems, the problem must first be transformed into several binary classification problems (Jair,2020).

- **Performance of SVMs in unbalanced datasets:** In case of unbalanced datasets, users face the issue of correct classification of minority class objects which is quite a challenging problem. Normal classification methods such as SVM do not work well for these skewed data sets because it is difficult to get the optimal separating hyperplane for an SVM trained with imbalanced data. Hence, due to this imbalancing problem, performance of the classifier is affected (Stamatatos,2010).

- **Choice of the appropriate kernel:** To solve a nonlinear problem, SVM leverages the kernel trick. This is equivalent to performing an embedding (projecting data to a different space where

data is linearly separable). However, an actual challenge during implementing SVM is that users are unknown of the space where data is linearly separable. Moreover, the kernel trick only works for a set of possible embedding. This means that there is no unanimous conclusion about which kernel is better or worse for specific applications. Users have to performed tests to identify the performance of SVM with different kernels, reaching the general conclusion (Kamran, 2015).

- **Selection of optimal hyperplane:** The optimal hyperplane is determined to maximise the generalisation ability of the model. However, even if the hyperplanes are properly selected, i.e., to maximise the space between classes, the classifier created may not have a high generalisation ability if the training set is not linearly separable. This is one of the challenging issue that users faced during implementing SVMs for the visualisation process.

- **Choosing between hard margin /soft margin:** Hard margin SVM is quite sensitive to outliers whereas soft margins SVM avoid iterating over outliers. The issue users faced is that soft margins SVM chooses a decision boundary that has non-zero training error.

- **Choosing the right value of C:** There are several parameters to optimise the SVMs. The C parameter is one very important factor that controls the over-fitting. To optimise this parameter while implementing SVMs becomes challenging for users. Usually this is done using grid techniques.

- **Probability estimates:** Probability estimates are not provided by SVMs directly rather these are calculated using an expensive five-fold cross-validation. It's part of the Python scikit-learn library's related SVC algorithm.

- **Sensitive to noisy data:** Another challenging issue users face is overfitting of data. This means that target classes are overlapping or the features can have similar or overlapping properties. This overlapping makes SVMs sensitive to noisy data.

Moving ahead with all these challenges in mind experienced by all types of users. Let's address how to overcome these challenges by the use of visualisations.

### 2.2.3 Overcoming challenges by creating visualisations

Different visualisation styles are available to gain insights from the data and decide the most appropriate from various machine learning algorithms. Such visualisations include:

1. **Histograms**
2. **Scatter plots**
3. **Heat maps**
4. **Pie charts**
5. **Confusion matrices**
6. **Correlation matrices etc.**

This section does not cover the in-depth detail of these visualisations. All of these methods are useful based on the context they are used in. The classification models are usually studied with the help of scatter plots, heat maps, and confusion matrices.

We will use these visualisations in our design methodology. In scatter plots, we plot each data point in 2-D or 3-D space, where its features are its dimensions. In a scatter plot, dots represent the values of two different numeric variables (also known as a scatter chart or scatter graph). The position of each dot on the horizontal and vertical axes indicates the value of each data point (Mike Yi, 2021).

Scatter plots depict the relationship between variables. Heat maps allow us to visualise different categories in the data in geographically indicated entities.

The use of a scatter plot will create an interactive visualisation that portrays existing classes in our data. These classes can be shaped differently with respect to it being either training data or testing data. This way the users are able to understand the misclassification of data if there are any. The confusion matrix will help us visualise how our SVM model is performing. The visualisation can also have an AUC curve graph, measuring the separability of the classes shown on the scatter plot. It tells how capable the model is to differentiate between classes.

#### 2.2.4 Summary

To summarise, the “background” chapter gave insights on the major components of SVM modelling for classification and how each aspect should be deeply understood to increase one’s explainability of the function. The challenges that were faced by all types of users were discussed and how there was research to back up these key difficulties that were faced on a daily basis. Many users still see these functions as a ‘black box’ model which doesn’t help them gain useful insights or rather they miss out on information while classifying their dataset. This is crucial when it comes to using it in real life, such as bioinformatics (cancer classification) or facial recognition. Subsequently, there are many ways to overcome these challenges, the prime solution as discussed is to use visual explanation of SVMs to the users. Not a still visualisation, but rather an interactive visualisation with useful plotting of graphs. Developers would be able to interpret the model with increased knowledge and competently improve their models. This visual explanation would not only help users understand the basic structure but also the mathematical function operating these functions. Next let’s take a look at existing visualisation tools that are already available for users to interact with.

## 2.3 Critical Analysis of existing visualisations

This part of the report contains critical analysis of one such interactive visualisation tool for SVMs.

The visualisations under study aim at teaching various aspects of SVM through easily understandable visuals.

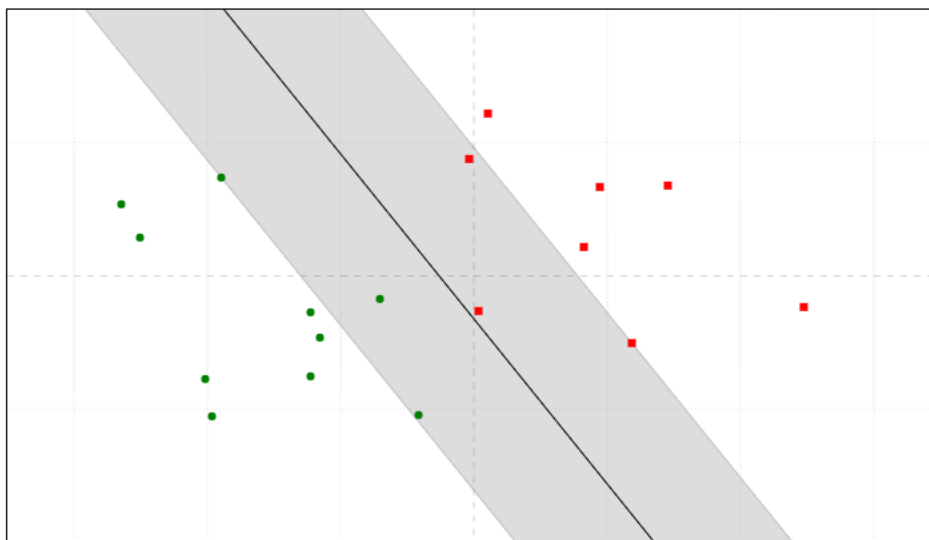
Link to the visualisation tool :- <https://jgreitemann.github.io/svm-demo>

### Interactive demo of Support Vector Machines (SVM)

February 12, 2018

tags: c++, machine-learning, svm, wasm

*Note: you may have to **disable your adblocker** for this demo to work.*



Toggle Clear Points  $\nu = 0.25$

Kernel: Linear  $\gamma = 1.0$   $c_0 = 0.0$

$$K(\mathbf{x}, \mathbf{y}) = \mathbf{x} \cdot \mathbf{y}$$

Figure 9: An interactive visualisation tool for SVMs.

### ***Key points regarding the interactive visualisation.***

The application has been created in C++. It displays the following functionality.

1. The application allows you to play with the dataset by adding a “dot” to the plane, this will **draw the hyperplane** with respect to the drawn dataset.
2. Any shape of data can be drawn with **manual clicking**.
3. There is an option to choose between a linear or nonlinear boundary.
4. **Support vector lines** are drawn along with the hyperplane.
5. The application provides the flexibility to choose between **different types of kernels** that could be used for the SVM model (Linear, RBF, Polynomial, Sigmoid).
6. Integration of **‘C’ parameter**. The users can play with the C regularisation parameter to improve the model.
7. Integration of the **gamma parameter**. The users can play with this hyperparameter to improve non-linear SVM models.
8. For the polynomial (quadratic) kernel, the app allows us to increase or decrease the **degree of the polynomial**.

### ***Critical Analysis***

The application is wonderfully made for all types of users to understand the basics of SVMs and how the hyperplane is drawn for different types of kernels. It majorly focuses on hand drawn data which gives flexibility to the user on the pace of learning. One could start with a low number of data-points then increase the number of data-points for both classes and analyse the results. The “Toggle” button helps users switch between one of the two classes (in this instance they are denoted by colour, green and red). Another great factor for hand drawn data is that the user can draw different patterns with an uneven amount of data-points from each class, and the hyperplane would still be

set accordingly. This technique induces increased imagination power of the user and they gain more insights as they try to play with the application.

The app also allows the user to learn how different types of kernels affect the classification in their data that they have drawn. With this they can also play with the hyperparameters and see its effect on the hyperplane. The support vector lines help identify the distance between the classes, so that the user would understand which would be the best fit model for them. Depending on the value of the hyperparameters the support vectors are taken in consideration for the hyperplane to be drawn (This was discussed in depth in the core components of SVMs section). These support vectors can be highlighted by checking the toggle button which reads “Highlight support vectors”. This way the users would know how the gamma parameter functions with regards to creating an improved model. Users could use hard and soft margins depending on the results that they want to acquire. Finally the app is able to clear all the data from the plane to restart again by clicking the “Clear Points” button.

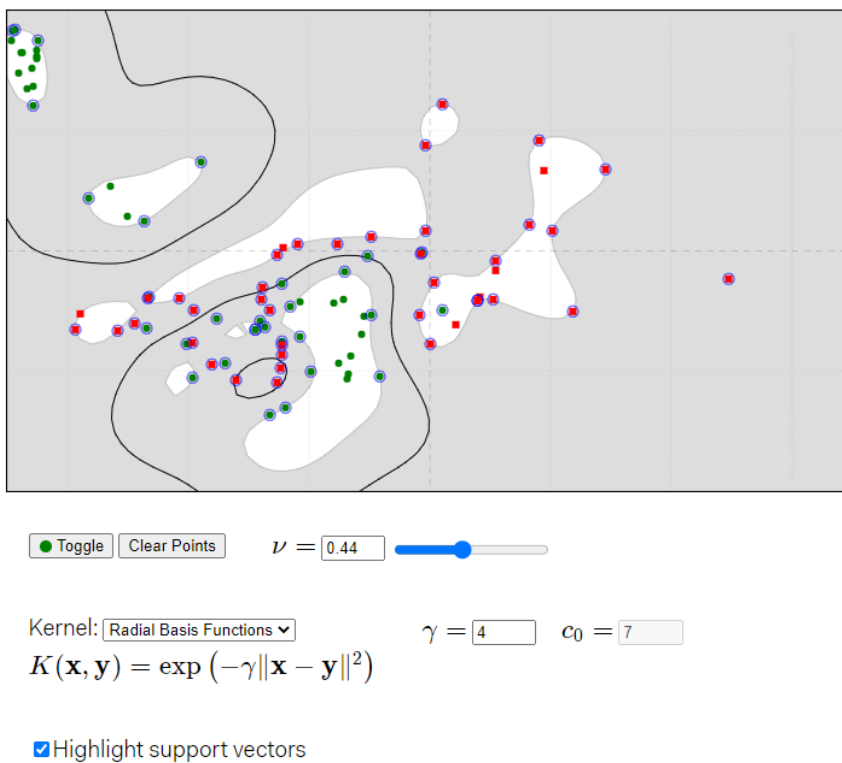


Figure 10 Showing the RBF function with highlighted support vector

Even though one would think that this visualisation tool would be the best for understanding SVMs. There are many aspects that are not covered within this tool that may cause the user some hindrance in studying SVMs. First is that the tool is only used for hand drawn datasets. That means that users cannot visualise their personal dataset or real datasets with real-life scenarios. Although the hand drawn dataset is more clean for beginners during their learning process, with time they would require to understand how it works when it comes to large scale datasets. The main disadvantage of this tool is that it only covers training data and not testing data. For users to understand the predicting side of SVMs (or any machine learning algorithm), it is mandatory to have some sort of testing data for the model. This would let users know how accurate their model is. One of the other reasons adding testing data would be useful is that it would help correctly adjust the hyperparameters to best fit the model. One of the other drawbacks with this tool is that there is no real-time measurement for the model. Meaning that adding graphs like confusion matrix, ROC curve in accordance with the current classification of data would be an added benefit, since it would give the users to comprehend their model with hard numbers. The tool is also sensitive to outliers when the dataset is very small. Visual tools are a great method of understanding algorithms but it is necessary to form a bridge between the visual aspect and the functional aspect that operates the algorithm. This is what is missing in this visual tool discussed above.

## 2.4 Conclusion and Future work

To conclude, the literature review gave a good insight on the working of SVMs and a critical analysis of one of the only good visual tools out there that comes close to being a great tool for users to indulge in and given them the explainability for the algorithm. It has shown the model behaviour keeping the major aspects of SVMs in mind. However, as discussed, there are major flaws in the visual tool that needs to be modified so that the tool becomes more efficient for explanation and



would reach out to a larger user base. Here are some few modifications that could be implemented on the future work of the visual tool :-

1. The visual tool should be able to evaluate large and more realistic models as compared to the one discussed before where the method is very simplified and feels rushed in an attempt to make the tool look more clean and understandable. However the “hand-drawn” data aspect of the tool is still a very niche implementation to increase the explainability of SVMs and should not be removed.
2. Implementation of evaluation methods such as confusion matrix and ROC analysis to evaluate the quality of the model (AMIA Annu Symp Proc. 2010). This will help users on the mathematical basis of the model and they would be able to evaluate the fitting of the model efficiently.
3. Tuning the sensitivity of the outliers in each kernel function is necessary. The above visual tool is not very accurate when it comes to data-points on a smaller scale. It tends to avoid noticing the outliers which underfits the model and therefore doesn’t provide the correct results. This could be frowned upon when programming the modified tool.
4. Training data, one of the important aspects of ML algorithms could be added in the modified visual tool. This implementation and with the use of evaluation graphs like confusion matrix would give the user major insights of the model and they could use it to their advantage in tuning the hyperplane with the given hyperparameters. This would also be advantageous in selecting the right kernel for any given dataset.
5. Finally, deep qualitative user evaluations could be performed which would help in acknowledging the fact if the interactive visualisation actually helps the user the way it should. This could be done by conducting a quantitative survey. The above literature review has not found any such user evaluation in any visual tools related to SVMs.

### 3. Requirements Analysis

This section will cover the functional and non-functional requirements of the interactive visual tool.

#### 3.1 Functional Requirements

The requirements of the project are solely based upon the research project questions discussed in the Introduction part. All the other functionalities that are added are not necessary. The main focus is to cover the requirements of the Minimum Viable Product(MVP) in this instance would be the application that is used for user evaluation. A MoSCoW approach has been used to rank priority.

<i>ID</i>	<i>Requirement</i>
FR-1	The application <b>must</b> allow the user to insert data with a pre-existing or a hand-drawn model.
FR-2	The application <b>must</b> generate a scatterplot with respect to the dataset that the user has given.
FR-3	The application <b>must</b> use the SVM algorithm on the dataset.
FR-4	The application <b>must</b> be interactive and reactive to the users commands.
FR-5	An option to change the kernel function <b>must</b> be available
FR-6	The user <b>must</b> be able to interact with the 'C' and gamma hyperparameter and be able to change its value.
FR-7	The application <b>should</b> be able to zoom in and out for clarity of the graph.
FR-8	A confusion matrix <b>must</b> be provided w.r.t the model.
FR-9	An ROC curve <b>must</b> be plotted w.r.t to the model.
FR-10	The application <b>could</b> have a noise slider to change the value of the noise in the data.
FR-11	The application <b>could</b> allow the user to change the testing data size.
FR-12	The application <b>could</b> have threshold frequency as a parameter.
FR-13	The user <b>should</b> be able to navigate through the dashboard with ease.

Table 1: Functional and non-functional requirements

#### 3.2 Non-Functional Requirement

<i>ID</i>	<i>Requirement</i>
NFR-1	The application's refresh time <b>must</b> be fast and the graph <b>should</b> render on a timely basis.

## 4. Design

This section covers the design of the visualisation of Support Vector Machine (SVM). Exploring various visualisation methods, finding which is appropriate for our application, and then move on to the design part.

### 4.1 Languages and Libraries

Since the visualisation will be created in python, let's briefly explore the most common python visualisation libraries. Following are the famous libraries for practically every data visualisation need—from static data plots to real-time interactive visualisations of the models.

**Matplotlib; Seaborn; Plotnine (ggplot); Plotly; Bokeh; pygal; geoplotlib; Folium**

Matplotlib is a comprehensive cross-platform library used for data visualisation and graphical plotting in python. It offers a viable open-source alternative to MATLAB. Developers can also use matplotlib's APIs (Application Programming Interfaces) to embed plots in GUI applications. It allows the users to create static, animated, and interactive visualisations and publication-quality plots in python. Seaborn is another python data visualisation library built on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics (Michael 2021). Matplotlib, seaborn, and ggplot are handy with rich documentation, so they are common among python visualisation learners. They are extensively used in data preprocessing and model selection procedures. But these libraries do not come with extensive interactive functionalities like Plotly. Since the aim is to build a sophisticated interactive visualisation for SVM, **Plotly is the best choice.**

## **Plotly**

Plotly is an open-source plotting library built in python that supports over 40 unique visualisation charts covering a wide range of statistical, geographic, scientific, and 3-dimensional use-cases. The biggest advantage of plotly over the libraries like matplotlib and seaborn is its interactive visualisations. Plotly allows the user to create beautiful interactive visualisations, where you can with different aspects of the display in real-time. With the help of the Plotly JavaScript library (plotly.js), beautiful interactive web-based visualisations could be created. These visualisations can be displayed in Jupyter notebooks. The visualisation will be created as a pure Python-built web application using **Dash**. Plotly works very well for non-web contexts, including desktop editors (e.g., PyCharm, QtConsole, Spyder) and static document publishing (e.g., exporting notebooks to PDF with high-quality vector images). Moreover, Plotly is more high-level than matplotlib and seaborn, etc., so it reduces the lines of code and makes it easy to remember them.

## **Dash Framework**

Dash is an open-source python framework created by Plotly for building interactive data-driven applications. It is built on top of Flask, Plotly.js, and React.js. There is no need to learn CSS, HTML, and Javascript to create interactive dashboards. You can create high-quality dashboards just by knowing python. The applications built using this framework are viewed on the web browser.

Dash gives us the full power of CSS functionalities. Every aesthetic element, including the sizing, colours, fonts, positioning, etc., of the app, is customizable.

Dash applications are built on the following two cores :

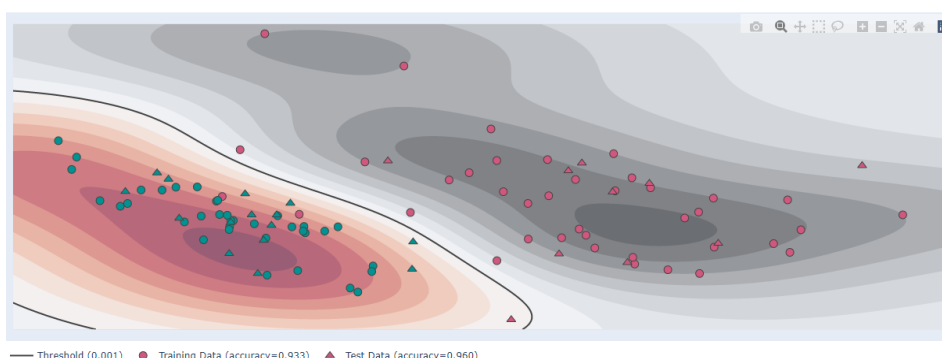
- Dash Layout
- Dash Callbacks

The dashboard's feel and appearance are defined by its layout. It includes Dash HTML components, which allow you to use Python to build and style HTML content such as images, headlines, and paragraphs. DASH is used to specify elements such as graphs, dropdowns, and so on, as well as their positioning, size, colour, and etc. Dash Core components are used to build graphs, dropdowns, and sliders, among other things. Callbacks are used to make dash apps more interactive. These are the functions that, for example, can define the action that occurs when a button or a dropdown is clicked.

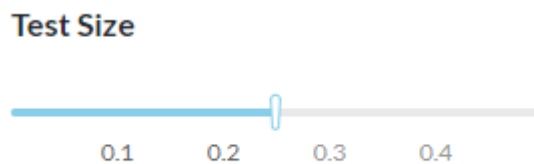
## 4.2 User Interface Design

Dash's layout and callback functions enable designing beautiful interactive user interfaces. These interfaces are dashboards with buttons, sliders, and dropdown menus. Dash is handy enough that you can bind a user interface around your Python code in an hour. It abstracts away all of the complexities and protocols in making user interfaces required to build an interactive web-based application. It enables anyone to build a beautiful front-end for your custom data analytics backend. The design of the application consists of nine components. They have been listed with a brief description in the following lines.

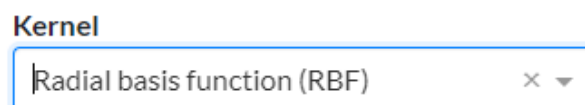
1. **Reactive Graph:** The largest component will be a reactive graph, comprising most of the place on the dashboard. This contains adding and removing the data within this graph. The hyper-plane and the support vector lines, drawn automatically in this graph, will change according to the dataset and the tuning of the hyperparameters. It will be a simple graph, as shown the the figure below.



2. **Different ways to add data:** The application allows three different ways to add data to the reactive scatter plot graph. The user can use the existing sci-kit learning dataset which is already provided in the application, or upload their own dataset or they could use the hand-drawn function and draw dots on the given area which would then convert into training and testing data-points on the graph. The figure is shown in [Appendix B](#).
3. **Testing and training Data:** There is a testing data slider integrated with all methods of adding data. This slider will dictate what percentage of the provided data will be testing data and the rest would be training data. See the figure below.



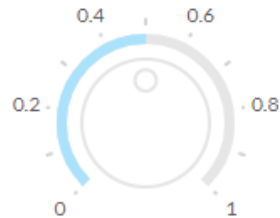
4. **Kernel Selection:** As discussed in the literature review this function to change the kernel is the most important. This drop-down menu allows the user to change the kernel function on any given dataset. See the figure below.



5. **Threshold:** There is a threshold knob added to the app. This knob is interactive and the user can set the threshold frequency from 0 to 1. This function increases or decreases the amount of data-points that are captured by the support vector machines. In case the user gets confused and due to some human error the plot is irrelevant due to incorrect change in the threshold, there is a “Reset Threshold” button integrated. This would reset the threshold and

the graph would be back to its initial stage. The default threshold frequency is different for different kernels.

#### Threshold



RESET THRESHOLD

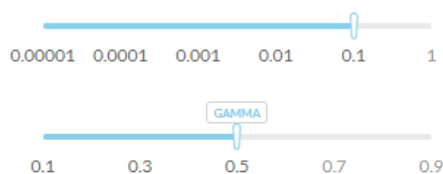
1. **Cost 'C' Slider:** The Cost slider is added to tune the SVM model by affecting the hyperparameter to control error. The value of the C regularisation parameter ranges from 0.01 to 10000. The second slider below changes according to the value of the first slide. This helps the user to gain precision in tuning the model.

#### Cost (C)

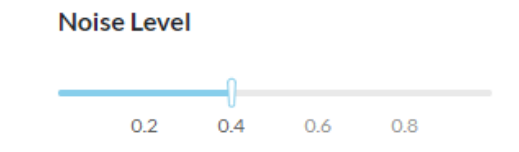


6. **Gamma Slider:** The gamma parameter slider is only unlocked when using a non-linear kernel function. The value of the gamma parameter ranges from 0.00001 to 1. The second slider changes according to the first slide, this gives the user more precision in tuning the model.

#### Gamma



7. **Noise Level Slider:** The app will allow the user to add noise to their data. Noise means adding the data belonging to one class in the area of another class, creating difficulty for the classifier to make accurate decision boundaries. This would challenge the application and the user would be able to learn the outcomes of changing of noise in their model.



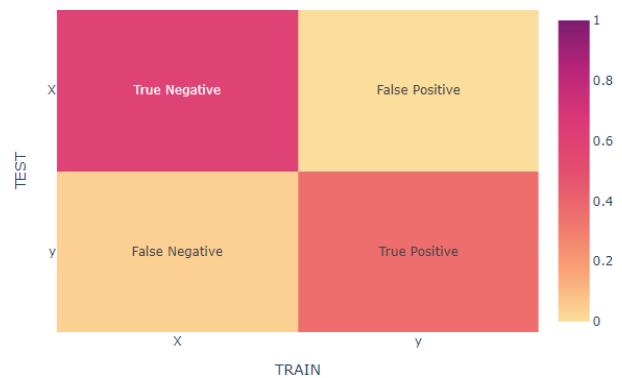
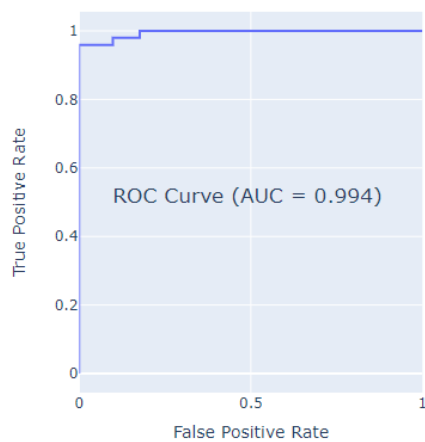
8. **Confusion Matrix**

Confusion matrix will also be displayed on the dashboard. A confusion matrix summarises prediction results on a classification problem that tells how many of the predictions are correct and incorrect with count values and broken down by each class. It tells you what your classification model is getting right and what types of errors it is making.

9. **AUC-ROC Curve**

The dashboard will also display the An evaluation metric is the Receiver Operator Characteristic (ROC) curve. It's a probability curve that separates the signal from the noise by plotting the True Positive Rate (TPR) versus the False Positive Rate (FPR) at various threshold values. The capacity of a classifier to discriminate between classes is represented by the Area Under the Curve (AUC). It's used to give a quick overview of the ROC curve. Higher the AUC means better the performance of the model at distinguishing between the positive and negative classes It looks like this.





### 4.3 Summary

This chapter aimed to communicate the design decisions relevant to the interactive visualisation of support vector machine algorithms. The section briefly overviewed the SVM algorithm and how it draws boundaries called hyper-planes to segregate the data into different classes. We then listed down several visualisation charts, mostly used in data analytics and machine learning. Different visualisations were analysed and chose the ones that would work best for describing the concepts of SVM interactively and jotted down the controls of the interactive visualisation will give us. Further explored different tools and libraries used for creating such applications and decided the DASH framework of Python's Plotly library would be the best fit for the interactive visualisation as it allows the user to create interactive web applications (dashboards). Then briefly described the components and design structure of the interactive dashboard and gave the details of how buttons, sliders, dropdown menus, or reactive graphs will let users play interactively with the SVM while understanding the black-box-natured concepts visually.

## 5. Implementation

This chapter covers the implementation process in detail; the challenges encountered during the implementation and how these were overcome; personal reflection on how improvements can be made in the implementation; and points of interest for future work to expand this application. To clearly present the work, the implementation section is split into three key areas of the application: back-end development; front-end development; and testing. This section contains the code snippets of different tasks, that collectively create an interactive visualisation of the support vector machine. This section might not include the low-level obvious code parts which are not necessary to be covered but walks the reader through the whole implementation so effectively that no ambiguity is left in the reader's mind.

### 5.1 Additional Resources

All the resources involved in the development of the visualisation have been submitted with this report. The linked GitHub repository contains these resources and also includes a comprehensive list of project dependencies and instructions. Moreover, a brief video demo of the interactive visualisation of SVM has also been uploaded in this repository.

LINK TO GITHUB REPOSITORY:- <https://github.com/PranavPai/Dissertation>

### 5.2 Development

While discussing the code, there will be jumping back and forth between back-end and front-end functions in the application code. Let's first talk about the backend and front end languages and frameworks used in the process, and then let's move on to the actual code.

### 5.2.1 Back-end

This sub-section covers the details about the back-end development of the interactive visualisation of support vector machines. This application has been primarily built in python, which is one of the most famous programming languages nowadays due to its extensibility and rich stack of awesome libraries. It's a general-purpose full stack programming language with extensive use in web development and the emerging technologies like Artificial Intelligence and Data Sciences. While developing the backend of the application it is necessary to first import certain packages and modules;

#### ***Importing packages for backend development:***

```
#=====
import time
import numpy as np
import pandas as pd

#=====
from utils.sampling import sampling, df_split, data_split
from utils.modeling import modeling
from utils.charting import prediction_plot, confusion_matrix_plot, roc_curve_plot
#=====
from utils.handle_func import *
from utils.split_components import *

#=====
```

The Python time package offers a variety of ways to express time in code, including objects, integers, and texts. It also has features other than expressing time, such as queuing during code execution and calculating your programme's efficiency (Alex,2020). NumPy is the most important Python package for scientific computing. It is a Python library that includes a multidimensional array object, derived objects (such as masked arrays and matrices), and a variety of routines for performing fast array operations, such as numerical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation, and much more

(NumPy,2022). Pandas is a Python library that provides quick, versatile, and powerful data structures for working with "relational" or "labelled" data. Its goal is to serve as the foundation for undertaking realistic, relevant data processing in Python. It also aspires to be the most powerful and versatile open source data analysis and manipulation tool available in any language (Pandas,2022).

### ***Input Processing***

The first step in the pipeline of generating an interactive visualisation was providing an SVM model at the backend with the input data. The SVM model made the decision boundary on the input the data, which belonged to different classes, after eating it along with the type of kernel function it will use and a bunch of other hyper-parameters.

First and foremost is the choice of input data fed into the SVM model. The app allows users to have three options regarding the choice of data:

1. To use existing predefined dataset.
2. To upload a new dataset externally.
3. To hand-draw the dataset in the assigned interface.

Predefined data refers to the dataset provided by the scikit-learn datasets collection. This is targeted for the users who do not have their own datasets created and organised and want to experiment with different aspects of the support vector machine to understand them. It is especially recommended for the beginners to experiment with the scikit-learn datasets for learning purposes and avoid the hassle of going through the hectic process of data creation and wrangling.

There are three predefined data styles that has been given in the dashboard :-

1. Moons
2. Linearly separable
3. Circles

It is recommended to experience each of them and see how the behaviour of SVM changes.

For the users, who have the luck of being able to create and organise their own datasets, they have an option to upload their datasets and experiment with the tool.

The last option is about hand-painting the data points on the assigned interface. This functionality is useful and powerful as it allows you to check the working of SVM on any complicated style of data. This functionality comes with a toggle button. Pressing the toggle button triggers a specific class of data whose data points you want to add in the interface now. After the choice of data comes the type of kernel function you want to use with the particular form of data you chose.

There are four types of kernel functions in the application:

1. Linear kernel
2. Sigmoid kernel
3. Polynomial kernel
4. Radial Basis Function (RBF) kernel

The following part of the code enables you to choose the kernel function.

```
kernel = html.Div([
    html.Strong('Kernel'),
    html.Br(),
    dcc.Dropdown(id={
        'type': 'svm_parameter',
        'index': 'kernel'
    },
        options={
            'rbf': 'Radial basis function (RBF)',
            'linear': 'Linear',
            'poly': 'Polynomial',
            'sigmoid': 'Sigmoid'
        },
        value='rbf',
        style={'width': '75%'}),
],
    style={'margin-bottom': '15px'})
```

## Hyperparameter setting

Then comes the setting of hyper-parameters 'C' and 'gamma'. The app uses sliders to control the parameter. The following function sets the cost 'C' parameter.

```
#setting cost parameter
cost = html.Div(
    [
        html.Strong('Cost (C)'),
        html.Br(),
        daq.Slider(id={
            'type': 'svm_parameter',
            'index': 'cost_power'
        },
            min=-2,
            max=4,
            value=0,
            marks={i: 10**i
                    for i in range(-2, 5)}),
        html.Br(),
        daq.Slider(
            id={
                'type': 'svm_parameter',
                'index': 'cost_coef'
            },
            min=1,
            max=9,
            value=1,
            step=1,
            handleLabel={
                "#showCurrentValue": True,
                "label": "COST"
            })
    ],
    style={'margin-bottom': '15px'})
```

Then you can increase/decrease noise in the data with the noise slider. Noise refers to the data in which the data points are put in a way that it's very difficult for the model to describe an accurate decision boundary between the classes of the data. The following code snippet implements the noise functionality in the backend.

```
noise_level = html.Div([
    html.Strong('Noise Level'),
    html.Br(),
    daq.Slider(id={
        'type': 'dataset_parameter',
```

```

        'index': 'noise'
    },
    max=1,
    value=0.4,
    step=0.1,
    marks={i / 5: i / 5
           for i in range(1, 5)})
],
    style={'margin-bottom': '15px'})\

```

## Callbacks

Callbacks are routines that are dynamically called by Dash whenever a property in an input component changes, in order to update another component's property (the output). These are the key players behind the interactivity of the Dash dashboards. They update the visualisations immediately according to the inputs you are providing. The dependence objects Input, State, and Output are always passed as positional arguments to `@app.callback` (either as positional arguments directly to `@app.callback` or as lists to the keyword parameters `inputs`, `state`, and `output`). The order of the positional arguments sent to the decorated callback function is determined by the order in which the dependence objects are provided. This indicates that the sequence in which the callback function arguments are defined is more important than their names. (Plotly,2022).

All HTML tags have classes in the Dash HTML Components (`dash.html`) module, and the keyword arguments explain HTML properties like `style`, `className`, and `id`. The Dash Core Components (`dash.dcc`) module generates higher-level components like controls and graphs. The Dash Core Components module (`dash.dcc`) can be imported and used with `dash import dcc` and gives you access to many interactive components, including dropdowns, checklists, and sliders. Dash DataTable is a table component that allows you to browse, edit, and explore huge datasets. DataTable uses standard, semantic HTML `table` markup, making it accessible, responsive, and simple to customise. This component was built from the ground up for the Dash community in React.js and Typescript. Its

API is user-friendly, and its behaviour may be totally customised using its properties (dash-table,2022).

```
from dash.exceptions import PreventUpdate
```

You may not want to change the callback output in some cases. This can be accomplished by raising a PreventUpdate module exception from dash.exceptions in the callback function.

### 5.2.2 Front-end

The front-end of the interactive visualisation has been created as a Dashboard. This dashboard has been developed in Dash Framework, which is an open-source python framework created by Plotly for building interactive data-driven applications. The applications built using this framework are viewed on the web browser. First it is necessary to import all packages required to create the front end of an interactive dashboard.

```
# -*- utf-8 -*-  
  
import dash_bootstrap_components as dbc  
from dash import (Input, Output, State, html, Dash, dcc, dash_table,  
                  get_asset_url, ALL, MATCH)  
from dash.exceptions import PreventUpdate  
#=====
```

These lines of code import the dash modules. Following lines contain a brief intro to these modules.

The first line in the above snippet imports a library 'dash\_bootstrap\_components' which is a collection of Bootstrap components for use with Plotly Dash, that makes it easier to build consistently styled Dash apps with complex, responsive layouts (dash-bootstrap,2020).



### ***Prediction Plot of SVM***

The model after taking the input data and other hyper-parameters, passes these to the function `prediction_plot(**kwargs)` which decides how the predicted decision boundary will look like in the pool of data points belonging to different classes.

### ***ROC curve***

Along with the prediction plot, an ROC curve is also embedded on the dashboard. An ROC curve (receiver operating characteristic curve) is a graph showing the performance of a classification model at all classification thresholds. The function `roc_curve` creates this ROC curve to see live performance of the model.

### ***Confusion Matrix***

There is also inclusion of a confusion matrix on the dashboard to see the performance of the model in the form of numbers. A confusion matrix is a way of summarising the performance of a classification algorithm. Classification accuracy alone can be deceiving if you have an imbalanced number of observations in each class or if your dataset has more than two classifications. Calculating a confusion matrix can assist you in determining what aspects of your classification model are correct and where it is incorrect.(Jason,2020).

The function below implements the confusion matrix on the dashboard.

```
def confusion_matrix_plot(**kwargs):  
    _data = kwargs['data']  
    split_data = _data[0]  
    model = kwargs['model']  
  
    #threshold of prediction plot  
    scaled_threshold = kwargs['threshold'] * (model[1].max() -  
                                              model[1].min()) + model[1]  
].min()  
    y_pred_test = (model[0].decision_function(split_data[1])) >
```

```

scaled_threshold).astype(int).astype(str)

#creating confusion matrix
#this will summarise the performance of app and algorithm
matrix = confusion_matrix(y_true=split_data[3], y_pred=y_pred_test)
mtx = matrix / matrix.sum()

label_text = [
    ["True Negative", "False Positive"],
    ["False Negative", "True Positive"]
]

fig = px.imshow(mtx,
                 x=['X', 'y'],
                 y=['X', 'y'],
                 color_continuous_scale='sunsetdark',
                 zmin=0,
                 zmax=1,
                 aspect="auto")
fig.update_traces(text=label_text, texttemplate="%{text}")

fig.update_layout(xaxis_title="TRAIN",
                  yaxis_title="TEST",
                  hovermode='closest',
                  transition=dict(easing='sin-in-out', duration=500
),
                  height=400,
                  margin=dict(l=10, r=20, t=40, b=20))

return fig

```

The above lines give a brief walkthrough of the code implementation. For complete understanding of the code and the working of each component in the program, the code has been well commented. The comments will let you grab each and every aspect very easily. Please visit the linked GitHub repository to access and experiment with the code.

## 5.3 Testing

The testing was divided into two parts to reduce risk and identify the challenges in the early stages of the project so that the final Minimum Viable Product (MVP) could be comfortably obtained. Early testing eliminates risks such as schedule delays and cost overruns caused by incomplete or substandard components. The less expensive a patch is, the earlier a defect is discovered in the

development process. The component testing for each functionality within the program, followed by full system testing upon completion.

### 5.3.1 Component Testing

Components here refer to functionalities within the application. The MVP directly or indirectly depends upon these components. During the implementation process, components were naturally separated and thoroughly tested to ensure their reliability. These elements are listed in Table 2.

Component	Description
Input Style	The experiment was conducted with each kernel function(the details have been provided in the above sections) and ensured the application correctly identifies each input form and proceeds efficiently.
Kernels & Hyper-parameters adjustment	Testing the kernels and hyper-parameters was a rigorous phase. This was conducted through a hit and trial approach and also used some established theories of hyper-parameter tuning of SVM to test the efficiency of the application.
SVM & decision boundary	The SVM model is at the heart of the application, it's vital to put the model's weights to the test by classifying sample inputs. This would assure that the model can effectively provide appropriate decision boundaries by providing accurate classifications.
Graphical interpretation	To offer the visualisations required by the MVP, the front-end depends on access to the data points and the decision boundary established by the SVM at the backend. To assure the correctness of results, it would be necessary to validate the Graph functionality's reliability.

Timely Updates/ Callbacks	The interactive online interface would demand rigorous testing of the callbacks in order to reduce problems and faults when the user study participants engage with the design probe.
------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Table 2: Different stages of component testing and its description

To perform these tests, the basic datasets were used that were provided by the scikit-learn library (as described in the previous sections), tested each component thoroughly and observed results. Testing was successful on each occasion.

### 5.3.2 System Testing

After having the component testing conducted successfully, the reliable results were evident of the whole application's efficiency and reliability due to the fact that each component is dependent on the one before it, a full system test of all interrelated components was performed to finish the testing phase and ensure the application's reliable capabilities. The findings showed that the programme was capable of receiving input data of any form along with the kernel choice and hyper-parameters and generating a front-end, interactive graph showing the data points and the decision boundary, along with the ROC curve and the confusion matrix on the UI.

## 5.4 Summary

This chapter has detailed thoroughly the implementation phase of the interactive SVM dashboard development. There were discussions on how the backend and the frontend of the application were developed and also jot down the main challenges experienced. The implementation followed an iterative approach, first the implementation of high-risk components and then testing it as to reduce risk, and then in subsequent iterations, the lower priority requirements.

While implementing the components, there were challenges while designing suitable data input options and integrating all of these in the main pipeline to manage the accurate display of the data

on the graph and creation of decision boundary. Implementing callbacks, despite their complex nature, was a tough part too. Although time intensive, these issues arose early in the implementation process due to their high priority and were easily resolved. To decrease risk and ensure the stability of the different modules of the system, thorough component testing was conducted throughout the iteration phases. Finally, system testing, which was carried out to guarantee the design probe's dependability for use in the user research, successfully tested the compatibility of all components.

#### 5.4.1 Critical reflection and future changes

Although the majority of the objectives were completed, the dash framework's data input methods and callbacks proved to be a significant drain on project resources. On second thought, the design phase may have delved a little more into these entities. This could help gain a better knowledge of their functionality before putting them in place. This would have saved a significant amount of time and work. Despite this, the tool gives a fantastic interactive representation of SVM for use in user research.

Work on expanding this application in the future should focus on adding a capability to have data compromising multiple classes and implementing this visualisation in 3-D. Moreover, future efforts may be directed towards including multiple input choices for the visualisation, which is a vital feature that skilled users wanting to diagnose and enhance models will find quite beneficial. Finally, the application could be made faster and resource efficient by implementing the data structures and callbacks effectively.

## 6. Evaluation

This chapter aims to evaluate the usefulness of our proposed interactive visualisation. It includes a survey conducted on a sample group's in-depth responses. This section describes the user study design, results and findings answer this project's research questions (stated below). Regarding previous work, the visualisation methods needed the element of interactivity to teach them effectively to the students and even intermediate geeks who do not have in-depth insight of how the SVM actually works on the low level. As a result of this insight, the strategy for the user study was carefully designed. This would provide valuable insight into SVM's interactive visualisation techniques, which have never been thoroughly investigated before, and might be used to spark theories in global ambition studies in the field. The two research questions for this project are listed in the table below.

1.	Does the interactivity of hyperparameters ('C', 'gamma') improve the model performance and reduce cognitive load on the subjects' minds over existing visualisation tools?
2.	Does the SVM visualisation tool containing any sort of interactivity such as, customizable data distribution, kernel types and other hyper-parameters increase the understandability of all types of users?

Table 1: Project research questions

### 6.1 Methodology

Interactive SVM visualisation tools like the one developed in this project are used for learning and teaching purposes where users typically are able to play with different aspects and parameters of support vector machine algorithms and see the changes in real time, which instils a deep understanding of the underlying concepts in their minds. As previously stated, the goal of this study is to generate in-depth qualitative user responses from a small sample of participants in order to address the project's research questions, with the application developed functioning as a design

probe in the process. With this insight, it was determined that a semi-structured online interview, in which users complete task-based objectives, would be the most effective technique for obtaining the required feedback.

### 6.1.1 Participant recruitment

Participants were chosen based on a set of criteria. To begin with, individuals had to be both students and over the age of 17. Second, users must fit into one of two user profiles: a novice learner with at least a basic understanding of computer science and machine learning basics, or an experienced user with prior expertise in support vector machine design and implementation. People who expressed interest in the study would be required to convey their level of expertise in order to be placed in one of these groups or rejected. Participants were ranked using the classification scale presented in Table 2.

Knowledge level	Description	User group
1	Has a basic understanding of machine learning concepts, but is unfamiliar with SVM.	Novice user
2	Has a clear grasp of machine learning but is new to SVM or knows very little about it.	Novice user
3	Has prior experience implementing SVM and has a strong knowledge of it.	Knowledgeable user
4	Has a firm understanding of SVM and deals with it on a regular basis.	Knowledgeable user

Table 2: During the recruitment process, categories were utilised to rank participant knowledge in the field.

People who fit into these categories are those who lack basic machine learning skills, which means they do not represent the general set of users who would interact with such a technology and are

thus unsuitable for this study. In order to address the second study question, it was critical to collect feedback from each of these user groups. Table 3 shows the user population that was enlisted. Due to the fact that the semi-structured interviews scheduled would run between 30 and 1 hour each, three users from each user group was regarded as a sufficient quantity within the scope of the project's resources.

Ref	Description	Knowledge level	User group
Participant 1	Has extensive knowledge of machine learning structure and algorithms and works with SVM at a research level.	4	Knowledgeable user
Participant 2	Has academic knowledge of SVM and has already deployed versions in the machine learning industry.	3	Knowledgeable user
Participant 3	Has a strong academic background in SVM and SVM models. Has worked with a variety of models in the past.	3	Knowledgeable user
Participant 4	Has a basic understanding of machine learning but struggles with SVM.	1	Novice user
Participant 5	Has a basic understanding of machine learning and has only had a rudimentary grasp of SVM in academia.	2	Novice user
Participant 6	Has understanding of machine learning and has only briefly learnt about SVM in academia.	2	Novice user

Table 3: Recruited participant descriptions and user profile groups



### 6.1.2 Ethical and legal considerations

When conducting research ethically, the involvement of human participants is taken into account and plans appropriately. There was no sensitive information collected in the study, such as addresses, emails, age, or name. Setting up video calls with the participants' permission for reference during the data processing stage. Participants were given a consent form prior to the study that explained what data was collected, why it was gathered, and what it's used for, and also their rights to withdraw and/or request deletion of their information at any time, and other study details such as the evaluator's contact information. Each participant was provided and signed a consent form, which may be provided in [Appendix C](#). In addition, the whole user research design was submitted for approval to the university's ethical board and was approved.

According to the Data Protection Act 2018 (DPR) and the General Data Protection Regulation, all data obtained from participants was anonymized, and they were offered the right to request to view or erase their data at any time (GDPR). It was also indicated that once the data was used, it would be removed.

### 6.1.3 Procedure

The semi-structured interviews were done in a specific order via an internet video conference with participants. Users were invited to provide comments on their experiences using both forms of visuals as stimulus. A Likert scale was used to collect minimal quantitative data in addition to open-ended questions. The user study was similar for both the novice learner group and the knowledgeable group, with the exception of a few open-ended questions.

Participants from both user profiles would be required to attend a 10 minutes presentation on the important concepts of SVM to begin. This would ensure that all participants had a similar

understanding of the fundamental concepts before being given the tasks. They would next interact with the two distinct SVM visualisations shown in Figure 11.

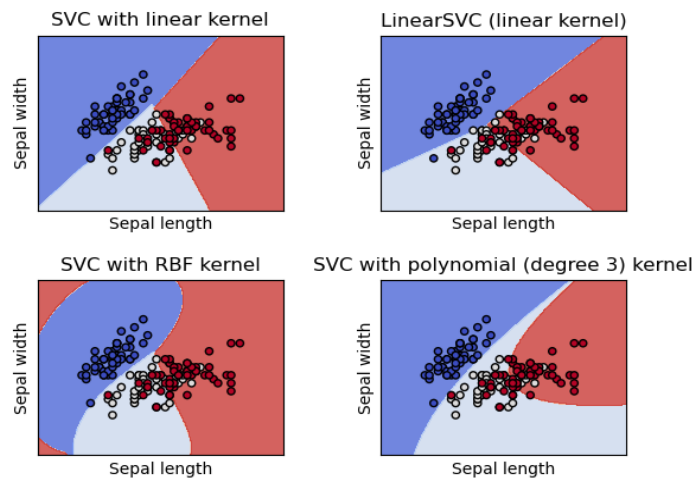


Figure 11 Non interactive SVM visualisation explaining different types of kernel functions



Figure 12 Interactive SVM visualisation

The non-interactive visualisation in Figure-11 represents different types of kernel functions being applied on the data and accordingly making changes. This visualisation is hard to comprehend and build a strong understanding of these difficult concepts. These are demonstrations of only kernel functions. The other parameters like 'C', 'gamma' etc. are even harder to absorb.

- The interactive visualisation in Figure-12 represents our SVM dashboard created in dash plotly. It has given interactive ability to all the aspects and parameters of SVM algorithm and allows the subjects to play and check each and every parameter of the algorithm and see the changes. Hence, it solidifies the concepts in the subjects' minds.

### ***Task 1 - analysing visualisations***

Task 1 sought to draw out responses to help investigate research question 1, and therefore related to the interactivity of SVM visualisation. To begin, participants were given access to the interactive visualisation and asked

*"Change the states and values of each parameter on the dashboard, whether a button, dropdown or a slider"*. This assignment was chosen because it closely resembles the types of interactions that typically occur when using such technologies. Users were then asked to fill out an abbreviated version of the NASA Job Load Index (TLX), a well-known questionnaire that has been carefully tested and shown effective in determining task workload (Hart & Staveland 1988). [Appendix D](#) contains the version used in this study. After that, participants were asked a series of open-ended questions about their impressions of the tool and which aspects of the assignment they found easy or challenging. Following that, participants were provided an overview and access to the visualisations, and the procedure was repeated.

### ***Task 2 - observational use***

Research question 2 is focused on the explainability of SVM via the interactive dashboard to users who want to better understand the models, whereas research question 1 is concerned with the efficiency of interactive visualisation of a dashboard. Task 2 entailed giving participants ample time to interact with the dashboard before they answered open-ended questions about their experience. Participants were given around 10 to 15 minutes to explore the dashboard and were free to make

comments throughout. Participants were asked follow-up questions that were specific to their user profile group. It was necessary to determine whether the visualisations helped new learners interpret the complexity of SVM functions and parameters, and whether their overall knowledge of SVM had improved since using the dashboard. To probe this, questions were asked such as "Can you describe the types of kernels and their impacts on the data?", and "After using the tool, has your knowledge improved? Why?".

The knowledgeable user group focused on their perspectives upon using the dashboard as a diagnostic tool to better understand and enhance models. "Would this dashboard be beneficial to you during the creation of your own SVM model?" participants were asked. "Do these visualisation tools help you comprehend why SVMs aren't as smart as they appear?". A detailed list of all questions asked can be found in the interview script in [Appendix E](#).

#### **6.1.4 Pilot test**

Each person who completed the original user study protocol contributed valuable suggestions for future improvements. This stage was crucial since it revealed portions of the script that needed to be improved. The removal of unrelated questions from the NASA TLX Questionnaire, the refinement of open-ended questions in the interview script, and improved communication of task instructions are just a few of the adjustments that resulted from this pilot test.

## **6.2 Results**

The feedback gathered from the interviews and quantitative results measured using the short questionnaire will help in supporting the findings of this user study.

### **6.2.1 Qualitative results**

The key parts of dialogue were extracted from the user during the evaluation process and aggregated for each participant. The user evaluation was a total of around 6 hours and it was out of this

project's resources to produce transcripts for each interview. Additionally, the analysis of the quotations drawn from the open-ended questions rendered the following key themes:

1. Maximum crucial information is unclear in static visualisations
2. Kernel functions remain in static visualisations
3. Gaining understanding about the hyper-parameters like 'C' and 'gamma' is difficult.
4. Understanding how the decision boundary alters its shape with the changing data is very hard
5. The interactive dashboard would be useful for understanding the underlying concepts.
6. Seeing the impact of different kernels and values of hyper-parameters on the decision boundary clarifies the ambiguities.
7. These interactive visualisations clearly portray the effect of noise on the data and decision boundary.
8. It clarifies the behaviour of SVM on different built-in shapes of data as well as it allows us to create our required shape of data with complete freedom. This teaches the SVM decision very efficiently.

### 6.2.2 Quantitative results

The results of the abridged NASA TLX questionnaire delivered during Task 1 are depicted in Figures 13 and 4. A lower score is preferable in all circumstances.

### Average NASA TLX Results for Task 1 (New Learner User Group)

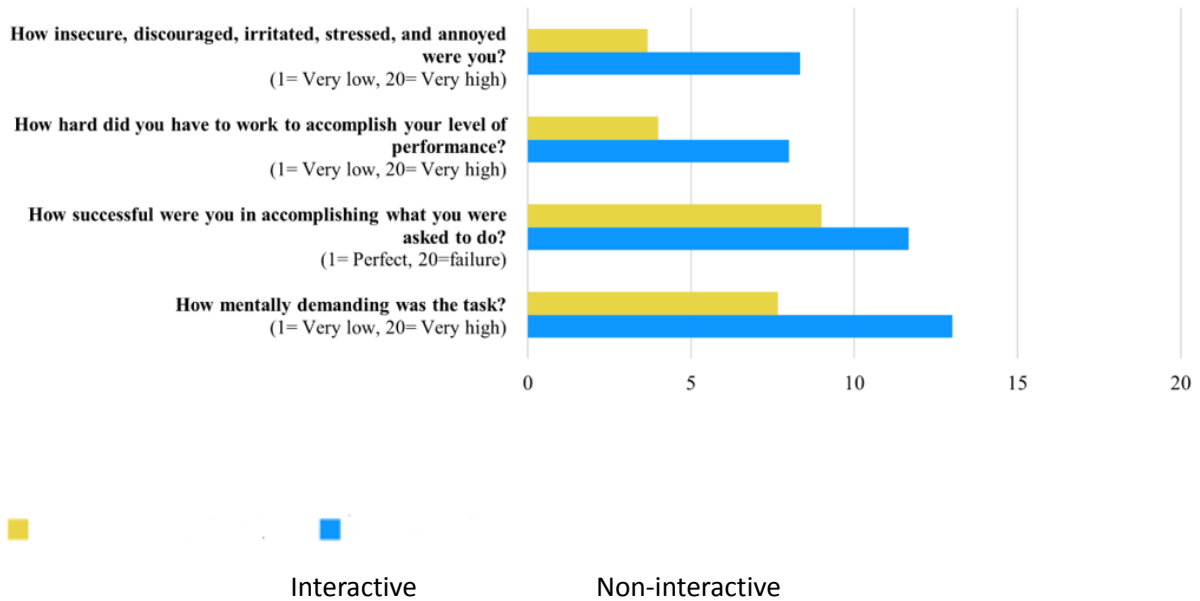


Figure 13: Results of the TLX Questionnaire for novoice users, average score

### Average NASA TLX Results for Task 1 (Knowledgeable User Group)

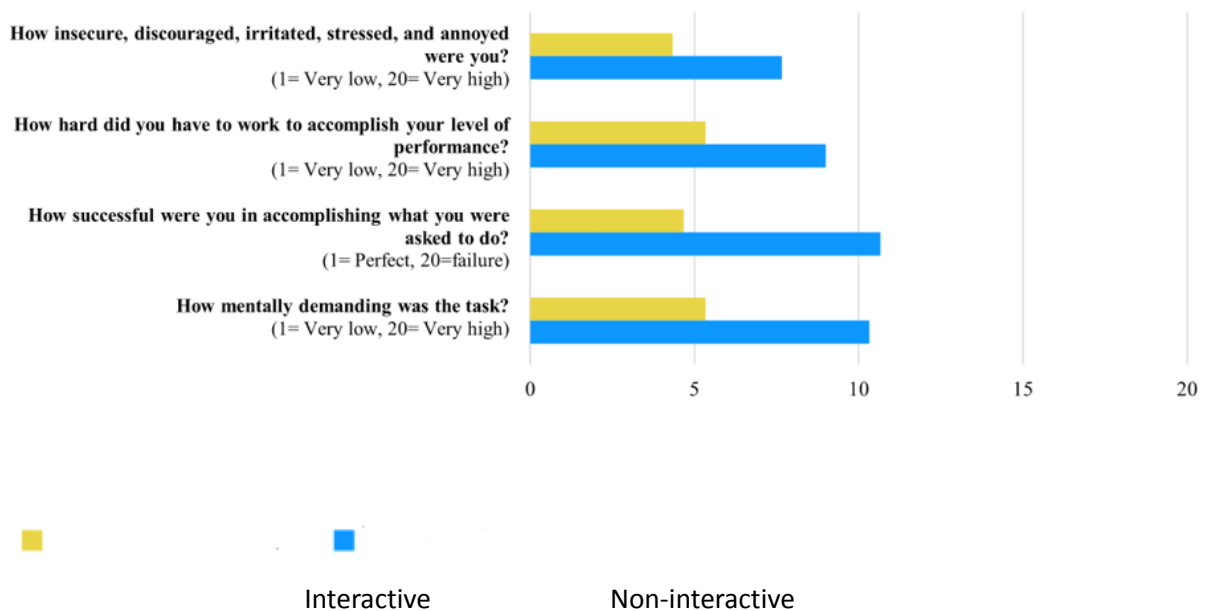


Figure 14: Results of the TLX Questionnaire for the knowing group, average score

### 6.3 Findings

**The interactive dashboard improved new learners' understanding of the SVM's underlying concepts.**

For the novice user group, the tool improved understanding for 3/3 of the participants. They managed to easily grasp the concepts of kernel functions, hyper-parameters 'C' and 'gamma' and noise in the data and their impacts on the decision boundaries. The third participant was rated as having knowledge level 1, signifying that they lacked the background to grasp the significance of these, implying that such a tool may be too difficult for persons with no prior knowledge of SVM. Positive feedback was offered to those whose knowledge had increased, and the advantages of using such a tool for learning were discussed.

*"It definitely helps as the underlying concepts of SVM are confusing so when we see the kernel functions, hyper-parameters 'C' and 'gamma' and noise in the data and their impacts on the decision boundaries, I can now tell what actually is happening."* -

*Participant 5*

*"I really didn't know what SVMs were in the beginning and now I know what kernel functions are" - Participant 6*

**Developers and professionals can benefit from the interactive SVM visualization's explainability.**

The knowledgeable user group unanimously agreed that the interactive dashboard would be beneficial to them and other machine learning engineers trying to improve model explainability. Participant 1, who has the highest level of knowledge in the group and works with SVM, stated that such a tool would be helpful even for experienced learners like himself to always revise their theories and keep them in check with the basics. The predictive model would also help senior managers who

are not really involved at the low level part of the project understand the algorithm in a simple and clean way.

*"It's absolutely good for testing myself from time to time and making sure I grasp how each notion of SVM works," says the author, who works in this field. - 1st participant*

*"Showing visualisations to stakeholders who are unfamiliar with SVM will assist me in explaining my job." - 1st participant*

In addition, the remaining two informed users provided favourable feedback. SVM models attempt to classify complicated huge datasets, according to participant 2. They explained that because the dataset is too complicated, employing a non-interactive visualisation for such use cases would not be appropriate in terms of explainability.

**The customizable data creation in interactive visualisation could be improved by adding capability for multiple classes data.**

The majority of the improvement suggestions were from experienced users who gave their thoughts on what types of innovative features they would want to see in such a tool. Although it was not in the scope of this research to implement anything beyond the two-class data for the design probe, it is important to provide these views since they reveal other desirable qualities from an SVM visualisation tool and may inspire future work in this area. Participants made it apparent that functionality for the user's choice of input was important, yet 4/6 participants also stated a need for more functionality. Participant 2 stated that they would like to be able to create data with many classes and have the tool display decision boundaries that are highly activated in that location.

"A setting that allows me to create data with several classes would be beneficial." - 2nd participant



## 6.4 Summary

The aim of this chapter was to conduct a detailed survey involving the different categories of subjects from new learners to the well-trained professionals. This user study introduced the users to the visualisations, conducted interviews and questionnaires and then communicated the results and findings. The planning, ethical considerations, recruitment phase, and user study technique are all described in full. Strong qualitative data was obtained from interviews stimulated by task-driven interactions with two SVM visualisations. The key themes from discussions supported three main findings, precisely: the interactive visualisation of SVM portray the difficult concepts like kernels and hyperparameters ('C', 'gamma') interactively, make them understandable and reduced cognitive load on the subjects' minds, they make the underlying concepts easy to grab and understand and the interactive SVM visualisation implementing interactively customizable data distribution, kernel types and other hyper-parameters provide improved explainability for all types of users including developers and experts. Gathered quantitative data using the NASA Task Load Index (TLX) Questionnaire in addition to open-ended question replies. The interactive dashboard was widely favoured in all categories, which helped to support the qualitative findings. Despite this, the quantitative results aren't large enough to be statistically significant, but they do show how the NASA TLX Questionnaire can be utilised in this situation to reliably evaluate workload, which is a vital metric for the interactive dashboard's performance.

## 6.5 Personal Reflection on Evaluation Design

There are various areas to reflect on and bring improvement. Regarding the design of the user study procedure, Table 4 lists down some key areas of improvement.

Task difficulty	During interviews, Task 1 took a long time to explain. This was most likely owing to my deep interest in the project, which caused me to see the task as being easier than it was. Future evaluations should develop tasks carefully and assess difficulty based on more extensive pilot testing.
Task length and participant concentration	The average interview lasted 30 minutes, but some lasted up to an hour. This causes problems since, as the session progresses, participant focus begins to fade, resulting in results that aren't representative of the average user. Interview times should be reduced in future reviews.
Alternating stimuli	In all six interviews, the non-interactive visualisation was shown before the interactive one for Task 1. This could lead to a bias in favour of the interactive visualisation because users are already familiar with some ideas and would expect the better tool to come in second. The stimulus in future evaluations should be varied.
Increased types of Data	This visualisation contest can be carried out in the future with an enhanced number of classes in the data. This would bring more amazing feedback from the subjects and drive more insights on the improvement end.

Table 4: Personal reflection about areas that could be improved upon

## 7. Project Conclusion and Future Work

The project has received positive feedback on its functionality and user explainability on SVMs. It has identified the gap in the current research of support vector machines and has implemented a visual tool to work upon these areas. The following are a summary to the research questions.

**Does the interactivity of hyperparameters ('C', 'gamma') improve the model performance and reduce cognitive load on the subjects' minds over existing visualisation tools?** - Yes, Task findings from the participants show that the parameters help in improving model performance, which as a part of a visualisation tool reduces cognitive load on subjects' minds over existing visualisation.

**Does the SVM visualisation tool containing any sort of interactivity such as, customizable data distribution, kernel types and other hyper-parameters increase the understandability of all types of users?** - Yes, the qualitative findings from the participants' interviews suggest that SVM visualisation tools containing interactivity are useful in offering clearer interpretability of SVMs from which they may learn, diagnose, and improve models for all types of users.

On a personal note, I would like to enhance this model into something more advanced. Where I would like to integrate more complex algorithms, more models that are known for its 'black box' structural complexity, into something very simple to learn using visual tools. This would improve explainability on all levels of data science. Interesting pre-existing datasets could exist that are particularly modelled to show the outliers of the dataset. This would make the user understand the drawbacks of each function as opposed to just understanding the working and the advantages of the function. Also the desired changes that were given by the participants has to be researched upon and implemented. This project has bought me an immense amount of inspiration on machine learning and creative methods of visualisation. This project at the end of the day also portrays how these could help understand complex learning algorithms in the field of computer science and that itself should be an inspiration to many who are keen on working on machine learning algorithms.

## 8. References

*A brief Introduction to Support Vector Machine.* [online] Available at: <https://towardsdatascience.com/a-brief-introduction-to-support-vector-machine-adf0f103a80f>

[online] Available at: <https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/>

*Support Vector Machine — Introduction to Machine Learning Algorithms.* [online] Available at: <https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47>

*Support Vector Machines(SVM) — An Overview.* [online] Available at: <https://towardsdatascience.com/https-medium-com-pupalerushikesh-svm-f4b42800e989>

IBM.com. *IBM Docs.* [online] Available at: <https://www.ibm.com/docs/it/spss-modeler/SaaS?topic=models-how-svm-works>

Analytics Vidhya. *SVM | Support Vector Machine Algorithm in Machine Learning.* [online] Available at: <https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/>

Analytics Vidhya. *SVM | Support Vector Machine | How does SVM work.* [online] Available at: <https://www.analyticsvidhya.com/blog/2021/03/beginners-guide-to-support-vector-machine-svm/>

Stecanella,B.,2017.[online]Availableat:<https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/>

Deepchecks.*Support Vector Machines | Deep Checks.* [online] Available at: <https://deepchecks.com/glossary/support-vector-machines/>

[online] Available at: <https://covidnewslive.com/beginners-guide-to-support-vector-machinesvm/>

*Working with high dimensional data.* [online] Available at: <https://medium.com/working-with-high-dimensional-data/working-with-high-dimensional-data-9e556b07cf99#:~:text=SVMs%20are%20well%20known%20for,than%20the%20number%20of%20samples.>>

*Demystifying Support Vector Machine—Part I.* [online] Available at: <https://medium.com/swlh/demystifying-support-vector-machine-part-i-b5b083844c9a>

2022. [online] Available at: <https://techvidvan.com/tutorials/svm-in-machine-learning/>

Analytics Vidhya. 2022. *Support Vector Machines | How is SVM better than Maximal-Margin & SVC.* [online] Available at: <https://www.analyticsvidhya.com/blog/2021/05/support-vector-machines/>

Kairimi,A.,2021. [online] Available at: <https://www.baeldung.com/cs/svm-hard-margin-vs-soft-margin>

Pham,T.,2020.[online]Available at:[https://www.researchgate.net/figure/Example-of-Radial-Basis-Function-RBF-kernel-mapping-data-from-non-linear-separable\\_fig12\\_281602651](https://www.researchgate.net/figure/Example-of-Radial-Basis-Function-RBF-kernel-mapping-data-from-non-linear-separable_fig12_281602651)

GeeksforGeeks. 2018. *Creating linear kernel SVM in Python - GeeksforGeeks.* [online] Available at: <https://www.geeksforgeeks.org/creating-linear-kernel-svm-in-python/#:~:text=Linear%20Kernel%20is%20used%20when,in%20a%20particular%20Data%20Set.>>

Ye, A., 2020. *Radial Basis Functions, RBF Kernels, & RBF Networks Explained Simply.* [online] Medium. Available at:

<https://medium.com/dataseries/radial-basis-functions-rbf-kernels-rbf-networks-explained-simply-35b246c4b76c>

Crsouza.com. 2010. *Kernel Functions for Machine Learning Applications* – César Souza. [online] Available at: <http://crsouza.com/2010/03/17/kernel-functions-for-machine-learning-applications/#sigmoid>

Lin, H., 2016. [online] Csie.ntu.edu.tw. Available at: <https://www.csie.ntu.edu.tw/~cjlin/papers/tanh.pdf>

Domínguez García, C., 2021. *Visualising the effect of hyperparameters on Support Vector Machines*. [online] Medium. Available at: <https://towardsdatascience.com/visualizing-the-effect-of-hyperparameters-on-support-vector-machines-b9eef6f7357b>

Karpathy, A., 2013. *svms Support Vector Machine in Javascript: demo*. [online] Cs.stanford.edu. Available at: <https://cs.stanford.edu/~karpathy/svmjs/demo/>

Stephan Dreiseitl, C. and ck, M., 2010. *Outlier Detection with One-Class SVMs: An Application to Melanoma Prognosis*. [online] PubMed Central (PMC). Available at: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3041295/>

Yi, M., 2021. *A Complete Guide to Scatter Plots*. [online] Chartio. Available at: [https://chartio.com/learn/charts/what-is-a-scatter-plot/#:~:text=A%20scatter%20plot%20\(aka%20scatter,to%20observe%20relationships%20between%20variables.>](https://chartio.com/learn/charts/what-is-a-scatter-plot/#:~:text=A%20scatter%20plot%20(aka%20scatter,to%20observe%20relationships%20between%20variables.>)

Medium. 2018. *Support Vector Machines(SVM) — An Overview*. [online] Available at: <https://towardsdatascience.com/https-medium-com-pupalerushikesh-svm-f4b42800e989>

Seaborn.pydata.org. 2021. *seaborn: statistical data visualisation — seaborn 0.11.2 documentation*. [online] Available at: <https://seaborn.pydata.org/>

Analytics Vidhya. 2017. *SVM | Support Vector Machine Algorithm in Machine Learning*. [online] Available at: <https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/>

Ronquillo, A., 2019. *A Beginner's Guide to the Python time Module – Real Python*. [online] Realpython.com. Available at: <https://realpython.com/python-time-module/#:~:text=The%20Python%20time%20module%20provide,s.the%20efficiency%20of%20your%20code>>

Numpy.org. 2022. *What is NumPy? — NumPy v1.22 Manual*. [online] Available at: <https://numpy.org/doc/stable/user/whatisnumpy.html>

Dash.plotly.com. 2022. *Flexible Callback Signatures | Dash for Python Documentation | Plotly*. [online] Available at: <https://dash.plotly.com/flexible-callback-signatures>

npm. 2022. *dash-table*. [online] Available at: <https://www.npmjs.com/package/dash-table>

Dash-bootstrap-components.open source.faculty.ai. 2016. *Dash Bootstrap Components*. [online] Available at: <https://dash-bootstrap-components.opensource.faculty.ai/>

Brownlee, J., 2020. *What is a Confusion Matrix in Machine Learning*. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/confusion-matrix-machine-learning/#:~:text=A%20confusion%20matrix%20is%20a,two%20classes%20in%20your%20dataset.>>

## 9. Bibliography

kumar, R., 2017. Signature Verification Using Support Vector Machine (SVM). *International Journal of Scientific Research and Management*,.

Jain, S. and Saha, A., 2021. Improving Performance by Genetically Optimising Support Vector Machine to Detect Code Smells. *SSRN Electronic Journal*,.

Wu, Q. and Zhou, D., 2005. SVM Soft Margin Classifiers: Linear Programming versus Quadratic Programming. *Neural Computation*, 17(5), pp.1160-1187.

Google Developers. 2022. *Classification: ROC Curve and AUC | Machine Learning Crash Course | Google Developers*. [online]

Saranli, A. and Baykal, B., 1998. Complexity reduction in radial basis function (RBF) networks by using radial B-spline functions. *Neurocomputing*, 18(1-3), pp.183-194.

Auria, L. and Moro, R., 2008. Support Vector Machines (SVM) as a Technique for Solvency Analysis. *SSRN Electronic Journal*,.



### 10.3 APPENDIX C - User Study Consent Form

#### **User Study Information Sheet - Visual Explanation of Machine Learning (SVMs) User Study Consent Form**

##### *What will the study involve?*

User feedback on various visualisation techniques on Support Vector Machines was gathered as part of the study (SVMs). Participants will be asked to participate in a fully online semi-structured interview in which they will interact with a variety of visualisation tools and provide feedback on their experience.

##### *Who can participate?*

- Heriot-Watt students or staff members over the age of 16.
- Participants must come from a computer science/mathematics background and ideally have some interest in AI/Machine learning.
- Knowledge of SVMs is not required.

##### *What data will be collected?*

- User feedback might be recorded (audio only) on MS Teams.
- Personal identifiable data such as name, age, address etc will not be collected.

##### *How will your data be used?*

The tools will be evaluated based on the responses and observations collected. In a written dissertation report, such responses may be quoted anonymously. If audio is recorded, it will only be utilised by the evaluator to refer back to the evaluation stage. They'll be removed after that.

##### *Further Information*

You may have the right to withdraw from the study at any point and your data to be permanently discarded and not used on the evaluation. You will have a constant point of contact throughout your study, you may email [ppp1@hw.ac.uk](mailto:ppp1@hw.ac.uk) at any point if you have any concerns or questions.

- ☐ I understand the stated terms and agree to participate in the study and have my data recorded and used in the evaluation stage of a dissertation project.

**Sign:**

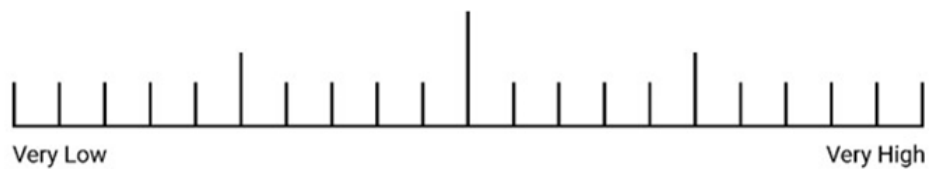
**Date:**



#### 10.4 APPENDIX D - NASA Task Load Index (TLX) Questionnaire

##### **Mental Demand**

How mentally demanding was the task?



##### **Performance**

How successful were you in accomplishing what you were asked to do?



##### **Effort**

How hard did you have to work to accomplish your level of performance?



##### **Frustration**

How insecure, discouraged, irritated, stressed, and annoyed were you?



## 10.5 APPENDIX E - Interview Script

### USER INTERVIEW SCRIPT

#### 1. Record data on how experienced users are with SVMs

- a. Question: Have you covered support vector machines at university?
- b. Question: Do you feel you could describe the basics of SVMs?
- c. Question: Have you used SVMs before?

#### 2. Introduce SVMs through a youtube video.

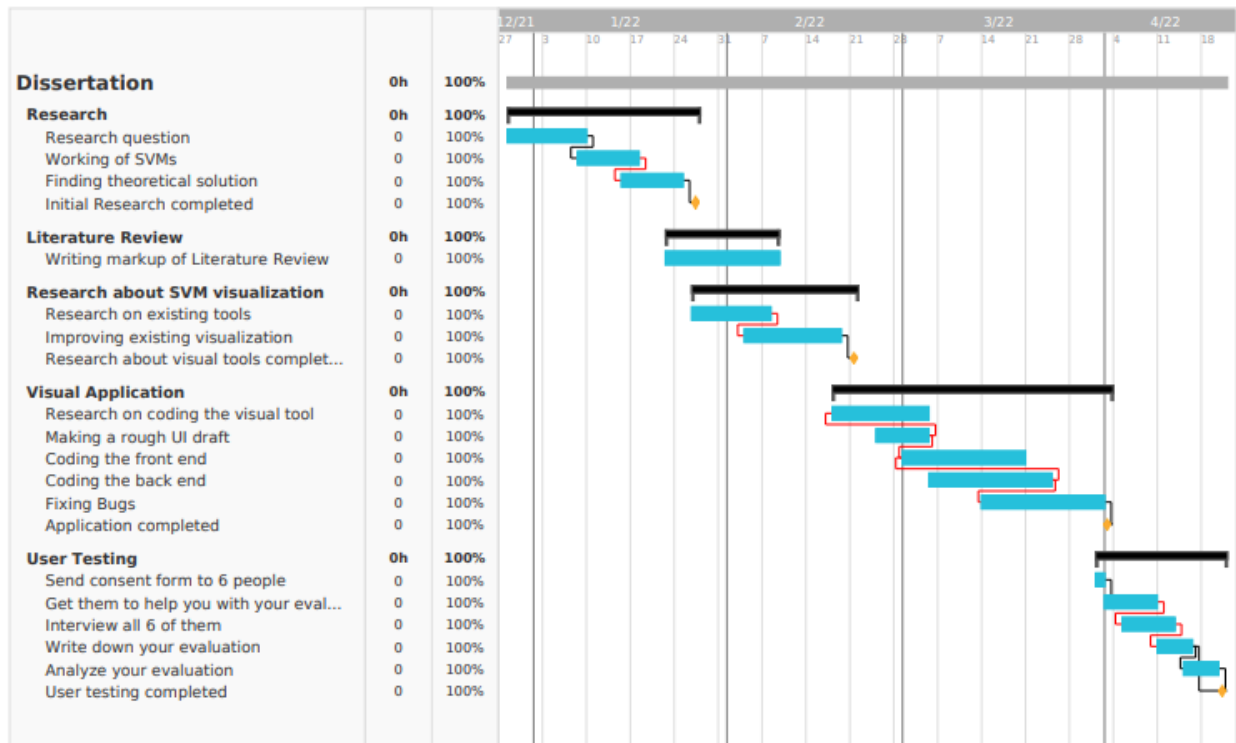
#### 3. Addressing RQ - 1

- a. Give users access to the visualisation tool
- b. Show the user the basics of how to use the tool to understand SVMs
- c. "You will now be asked to draw your own set data and by using RBF kernel, try to find the best fit model by tuning the hyperparameter"
- d. You will now interact with the visualisation tool
- e. Record the time taken
- f. NASA TLI questionnaire
  - How mentally demanding was the task?
  - How was the pace of the task?
  - How successful were you in accomplishing the task
  - How hard did you have to work to accomplish the task?
  - How insecure, discouraged, irritated, stressed, and annoyed were you?
- g. Question: How do you think the interface could be improved to help with the task?
- h. Question: What aspects did you find easy or difficult during the task? Could you elaborate?
- i. Show users a non-interactive web tool.
  - Ask the same questions as above.

#### 4. To address RQ-2

- a. Explain the model in context
- b. Question: Do you understand how these hyperparameters work?
- c. Question: After using this tool, how has your knowledge on SVMs either improved or did you get more confused.
- d. Question: Why is it like that? Could you elaborate?
- e. Question: What aspect did you understand better?
- f. Question: What aspects did you not understand, if any?
- g. For an advanced user. Does it make sense that SVM are not as complicated as they are portrayed?

## 10.6 APPENDIX F - Gantt Chart



### 10.7 APPENDIX G - Declaration Form from University

Course code and name:	F20PA - Research Methods and Requirements Engineering
Type of assessment:	Individual
Coursework Title:	Dissertation (D2)
Student Name:	Pranav Pai
Student ID Number:	H00310296

**Declaration of authorship. By signing this form:**

I **declare** that the work I have submitted for individual assessment OR the work I have contributed to a group assessment, is entirely my own. I have NOT taken the ideas, writings or inventions of another person and used these as if they were my own. My submission or my contribution to a group submission is expressed in my own words. Any uses made within this work of the ideas, writings or inventions of others, or of any existing sources of information (books, journals, websites, etc.) are properly acknowledged and listed in the references and/or acknowledgements section.

I confirm that I have read, understood and followed the University's Regulations on plagiarism as published on the [University's website](#), and that I am aware of the penalties that I will face should I not adhere to the University Regulations.

I confirm that I have read, understood and avoided the different types of plagiarism explained in the University guidance on [Academic Integrity and Plagiarism](#)

**Student Signature** (type your name): *Pranav Pai*

**Date:** 21/04/2022

