Support Vector Machines

Luis Rojo-González[a,b]

[a]*Department of Industrial Engineering, Universidad de Santiago de Chile, Chile*
[b]*Department of Statistics and Operational Research, Universitat Politècnica de Catalunya, Spain*

Abstract

Support Vector Machines are studied by several researches taking an important role in different applications due to its flexibility and variation to do classification as well as classification depending on a parameter that allow us to work with that kind of data which can be not so separable. On this way, we study the mathematical model in both primal and dual formulation using Wolfe duality to get it. One we state the problem and having the conditions to compute the original variables (i.e. primal variables) we apply it on two different dataset to study its behavior on running times and error considering a split of 75% of the set to calibrate (train) the model and the other 25% to verify (test) how much accurate is on the data which is not considered before. Those dataset come from a pseudo-random number generator with 1000 observations and another dataset that has the information about a product that is purchased by a customer or not according to its age and income and are solved using AMPL/CPLEX. Also, this process is repeated but using the scikit-learn package of Python with the same purpose. Obtained results show that the algorithm used by Python is quite close to the optimal solution but with an important gain of computational (running) time. [1]

1. Introduction

Support Vector Machines (SVM) is a generalization of linear decision boundaries for classification where it can be possible to find a separable case as well as non-separable case for classified data [1]. This Machine Learning (ML) technique has been widely studied and applied in several researches from many areas as medicine [6, 8], finance [12, 5], sports [11], among other. Its development has been really important such that it has been developed variations of it such as Support Vector Regression (SVR) [10], Support Vector Networks (SVN) [4] and Support Vector Clustering (SVC) [2]. In this work, we study SVM on first place as a Quadratic Programming (QP) model on its primal formulation finding the dual formulation applying Wolfe duality [13] due to primal formulation is a quadratic optimization problem. Once we just have defined both mathematical formulations, computational experiments are performed on two datasets to show their behavior and obtained results.

2. Statement of the problem

Let $\mathcal{I}$ be a set of observations and $\mathcal{J}$ be a set of attributes. Thus, $\mathcal{X}$ defines a matrix of $i \times j$-dimension $\forall i \in \mathcal{I}, j \in \mathcal{J}$. And a (hyper) plane on its general form $\beta_0 + \sum_{j \in \mathcal{J}} x_{i,j}\beta_j$, $x_{i,j} \in \mathcal{X}$, $\forall i \in \mathcal{I}$, $j \in \mathcal{J}$. Let $y_i = \{-1, 1\}$ the classification of observation $i \in \mathcal{I}$ and its obtained error from classification process e_i, $\forall i \in \mathcal{I}$. Finally, consider $C \in \mathbb{R}_+$ a parameter which represents the weight of obtained errors.

Email address: luis.rojo.g@usach.cl (Luis Rojo-González)

[1] You can find AMPL and Python implementations at: https://drive.google.com/drive/folders/1Yepwlyp3nBTNwFaUuYqsv_1-lggYLDmR?usp=sharing

3. Support Vector Machines: Mathematical Formulation

Consider the elements stated above in Section 2 and the classifications rule defines as follows we can formulate a mathematical model to get the solution for the required (hyper) plane which could separate the data in two possible classifications according to vector y_i. Also, starting on QP model we can get its dual form by using Wolfe Duality due to with it we will be able to use a transformation of the data known as *Kernel* [9, 3].

3.1. Quadratic Programming (QP): Primal Formulation

The goal of the model is to find kind of rule that separates the data in two groups such as we stated above. Also, due to we could have some datasets that are completely separable we have to consider an error term penalized by a constant (C) which allows us incorporate it to our formulation and *to control* the obtained plane. Despite of we can separate the data we have to define a rule that really classify the data according to some criteria. On this way, we define the rule stated as follows

Definition 1. ***Classification rule.***
Let $G(x)$ a function that classifies points defined as follows

$$G(x_i) = sign\left(\beta_0 + \sum_{j\in\mathcal{J}} x_{i,j}\beta_j\right) \quad \forall i \in \mathcal{I} \tag{1}$$

Equation (1) *means that if $G(x_i) < 0$ observation i belongs to classification represented as -1, otherwise it belongs to classification $+1$.*

Then, the (hyper) plane must be necessary to compute $G(x)$ is obtained from the mathematical model

$$min \quad \frac{1}{2}\sum_{j\in\mathcal{J}} \beta_j^2 + C\sum_{i\in\mathcal{I}} e_i \tag{2}$$

$$s.t. \quad y_i\left(\beta_0 + \sum_{j\in\mathcal{J}} x_{i,j}\beta_j\right) \geq 1 - e_i \qquad \forall i \in \mathcal{I} \tag{3}$$

$$e_i \geq 0 \qquad \forall i \in \mathcal{I} \tag{4}$$

Note that $y_i\left(\beta_0 + \sum_{j\in\mathcal{J}} x_{i,j}\beta_j\right) \geq 1 - e_i$ can be written as $y_i G(x_i) \geq 1 - e_i$, which means that this constraint represents those predictions which are in wrong *side* (i.e. those points that are wrongly classified), $\forall i \in \mathcal{I}$.

3.2. Quadratic Programming (QP): Dual Formulation

On the same way that we stated the Primal Formulation of SVM we can define its Dual Formulation due to we would apply Kernels to our data which would bring us some advantages when they are not completely separable.

Starting on Primal Formulation we have the Lagrange relaxation (L) with dual variables α_i and μ_i defined as

$$L = min\left(\frac{1}{2}\sum_{j\in\mathcal{J}} \beta_j^2 + C\sum_{i\in\mathcal{I}} e_i\right) - \sum_{i\in\mathcal{I}} \alpha_i\left(y_i\left(\beta_0 + \sum_{j\in\mathcal{J}} x_{i,j}\beta_j\right) - (1 - e_i)\right) - \sum_{i\in\mathcal{I}} \mu_i e_i \tag{5}$$

Then, setting the respective derivates to zero, we get

$$\beta_j = \sum_{i\in\mathcal{I}} \alpha_i y_i x_{i,j} \quad \forall j \in \mathcal{J} \tag{6}$$

$$0 = \sum_{i \in \mathcal{I}} \alpha_i y_i \tag{7}$$

$$\alpha_i = C - \mu_i \quad \forall i \in \mathcal{I} \tag{8}$$

as well as the positivity constraints $\alpha_i, \mu_i, e_i \geq 0$, $\forall i \in \mathcal{I}$. By substituting (6)-(8) into (5), we obtain the Lagrangian (Wolfe) dual formulation

$$max \qquad \sum_{i \in \mathcal{I}} \alpha_i - \frac{1}{2} \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{I}} \alpha_i \alpha_j y_i y_j K_{i,j} \tag{9}$$

$$s.t. \qquad \alpha_i \leq C \qquad \forall i \in \mathcal{I} \tag{10}$$

$$\sum_{i \in \mathcal{I}} \alpha_i y_i = 0 \tag{11}$$

$$\alpha_i \geq 0 \qquad \forall i \in \mathcal{I} \tag{12}$$

and to recover the primal variables we have, applying the Karush-Kuhn-Tucker (KKT) conditions

$$\alpha_i \left(y_i \left(\beta_0 + \sum_{j \in \mathcal{J}} x_{i,j} \beta_j \right) - (1 - e_i) \right) = 0 \quad \forall i \in \mathcal{I} \tag{13}$$

$$\mu_i e_i = 0 \quad \forall i \in \mathcal{I} \tag{14}$$

$$y_i \left(\beta_0 + \sum_{j \in \mathcal{J}} x_{i,j} \beta_j \right) - (1 - e_i) \geq 0 \quad \forall i \in \mathcal{I} \tag{15}$$

where primal variables β_j, β_0 can be computed as

$$\hat{\beta}_j = \sum_{i \in \mathcal{I}: \hat{\alpha}_i > 0} \hat{\alpha}_i y_i x_{i,j} \quad \forall j \in \mathcal{J} \tag{16}$$

$$\hat{\beta}_0 = \frac{1}{y_i} - \sum_{j \in \mathcal{J}} \hat{\beta}_j x_{i,j} \quad \forall i \in \mathcal{I} | 0 < \alpha_i < C \tag{17}$$

Also, we can define a rule that classifies our data according to obtained results such as follows

Definition 2. ***Classification rule.***
Let $\hat{G}(x)$ a function that classifies points defined as follows

$$\hat{G}(x_i) = sign \left(\hat{\beta}_0 + \sum_{j \in \mathcal{J}} x_{i,j} \hat{\beta}_j \right) \quad \forall i \in \mathcal{I} \tag{18}$$

Equation (18) *means that if $\hat{G}(x_i) < 0$ observation i belongs to classification represented as -1, otherwise it belongs to classification $+1$.*

Table 1: Resolution time (s) for QP model and errors according to C.

		Primal			Dual		
	C	Training error	Validation error	Time (s)	Training error	Validation error	Time (s)
1	0.5	0.0613	0.088	0.1202	0.0613	0.088	1.3931
2	1	0.0627	0.084	0.1737	0.0627	0.084	2.7563
3	5	0.0653	0.084	0.2279	0.0653	0.084	4.1670
4	10	0.0587	0.080	0.2831	0.0587	0.080	5.4660
5	50	0.0613	0.080	0.3522	0.0613	0.080	6.6166
6	100	0.0613	0.080	0.4137	0.0613	0.080	7.7121
7	1000	0.0600	0.080	0.4609	0.0600	0.080	8.8092
8	5000	0.0600	0.080	0.5198	0.0600	0.080	9.9146
9	10000	0.0600	0.080	0.5747	0.0600	0.080	11.0654

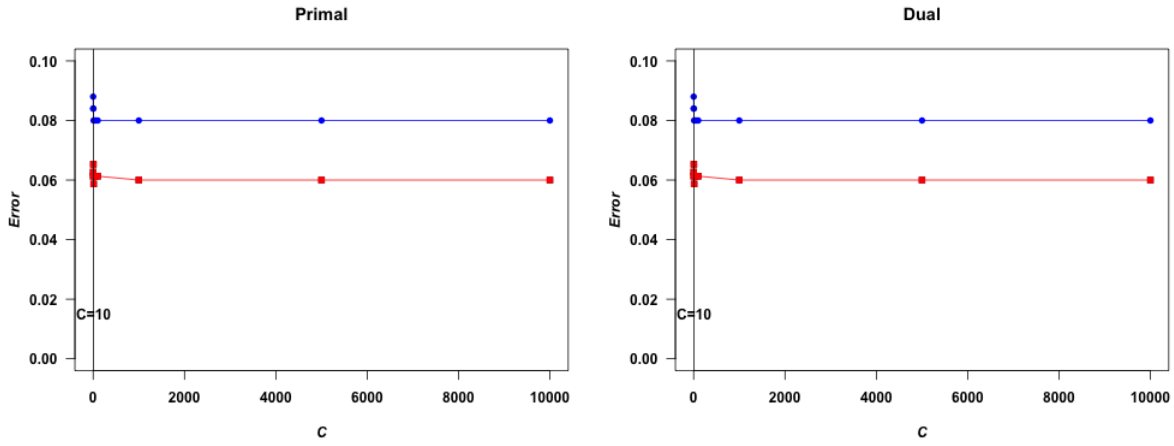

Figure 1: Error of primal and dual model according to C in training set (red line) and test set (blue).

4. Computational experiments

To observe the model's behavior we propose to study two datasets, one provided for a generator of pseudo-random numbers with four independent variables and another dataset with two independent variables: x_1: Age and x_2: Estimated Salary, which response variable is affirmative if the client buy a product.

The implementation of the mathematical model was in AMPL version 20180110 and was solved using solver CPLEX 12.8.0.0 for integer programming whereas the implementation of scikit-learn was in Python 3.7. It was executed in a Macbook Air Intel Core i5, 1.6 Ghz, 8GB RAM.

For this, we will use both primal and dual formulation tuning parameter C and obtaining results from an optimal solution (provided by AMPL using CPLEX) and an approach (provided for Python with *scikit-learn* package [7]).

4.1. Pseudo-random number generator

We work on pseudo-random number generator first to see how is the training error of both formulations. Thus, Table 1 shows obtained results according to tuning parameter C considering first 750 observations over a total of 1000, also Figure 1 shows the behavior of error in both training sample and test sample and indicates that C in which training error is minimum.

As we can see both errors are equal in primal and dual formulation which in fact are constant while C increases, it means that it does not matter how much C increases the obtained results would be the same and also the most important analysis is that dataset is not *so* separable.

Table 2: Hyperplanes according to C.

		Primal					Dual				
	C	β_0	β_1	β_2	β_3	β_4	β_0	β_1	β_2	β_3	β_4
1	0.5	-6.373	3.3955	3.0148	3.2372	2.9665	-6.373	3.3955	3.0148	3.2372	2.9665
2	1	-7.3523	3.9376	3.4801	3.5581	3.5619	-7.3523	3.9376	3.4801	3.5581	3.5619
3	5	-8.7266	4.6985	4.0855	4.2491	4.2122	-8.7266	4.6985	4.0855	4.2491	4.2122
4	10	-9.0583	4.8342	4.2605	4.4614	4.3494	-9.0583	4.8342	4.2605	4.4614	4.3494
5	50	-9.4195	5.0406	4.4094	4.6644	4.5506	-9.4195	5.0406	4.4094	4.6644	4.5506
6	100	-9.4195	5.0406	4.4094	4.6644	4.5506	-9.4195	5.0406	4.4094	4.6644	4.5506
7	1000	-9.5316	5.1026	4.4544	4.7074	4.6361	-9.5316	5.1026	4.4544	4.7074	4.6361
8	5000	-9.5316	5.1026	4.4544	4.7074	4.6361	-9.5316	5.1026	4.4544	4.7074	4.6361
9	10000	-9.5316	5.1026	4.4544	4.7074	4.6361	-9.5316	5.1026	4.4544	4.7074	4.6361

Table 3: Resolution time (s) for QP model and errors according to C.

		Primal			Dual		
	C	Training error	Validation error	Time (s)	Training error	Validation error	Time (s)
1	0.5	0.1400	0.41	0.0682	0.1400	0.41	0.1743
2	1	0.1333	0.36	0.1120	0.1333	0.36	0.3771
3	5	0.1467	0.30	0.1494	0.1467	0.30	0.5451
4	10	0.1467	0.27	0.1909	0.1467	0.27	0.7202
5	50	0.1400	0.27	0.2308	0.1400	0.27	0.8909
6	100	0.1400	0.26	0.2697	0.1400	0.26	1.0639
7	1000	0.1400	0.26	0.3093	0.1400	0.26	1.2588
8	5000	0.1400	0.26	0.3473	0.1400	0.26	1.4250
9	10000	0.1400	0.26	0.3973	0.1400	0.26	1.5919

On the other hand, we just have some clues related to the accomplishment of strong duality due to obtained errors but get the variables results must be important to support this claim. Such as Table 2 we have the same variables' results, so we can claim that the strong duality is verified in this case.

Now, using Python library scikit-learn we have similar results according to the same dataset but with a considerable reduction of running time. Also, as we already know the optimal solution for each proposed parameter setting (C) we will just show the best solution of that one obtained from this process with its whole running time. Thus, the obtained hyperplane for primal and dual formulation are $-9.5316+5.1026x_1+4.4544x_2+4.7074x_3+4.6361x_4$ and $-9.5316+5.1026x_1+4.4544x_2+4.7074x_3+4.6361x_4$. Also, their error respectively are 6.8% for primal and 6.5% for dual formulation, respectively. Those results are possible to get with $C = 0.5$ for primal and C=1 for dual formulation. Finally, if we compare those results with those obtained from the optimal solution we have that for primal formulation with we get the same conclusion related to C but with an error of 6.13% in both cases (remind that the strong duality is verified), but if we compare the dual formulation where we have discrepancies results do not change a lot, i.e. the error of optimal solution is 6.27%. Anyway, running times decrease from 3.1202 to 0.0455 in primal formulation and from 57.9003 to 0.1839 (comparing the best case in python's results).

4.2. Classification of purchase a product or not

We work on pseudo-random number generator first to see how is the training error of both formulations. Thus, Table 3 shows obtained results according to tuning parameter C considering first 750 observations over a total of 1000, also Figure 2 shows the behavior of error in both training sample and test sample and indicates that C in which training error is minimum.

As we can see both errors are equal in primal and dual formulation which in fact are constant while C increases, it means that it does not matter how much C increases the obtained results would be the same

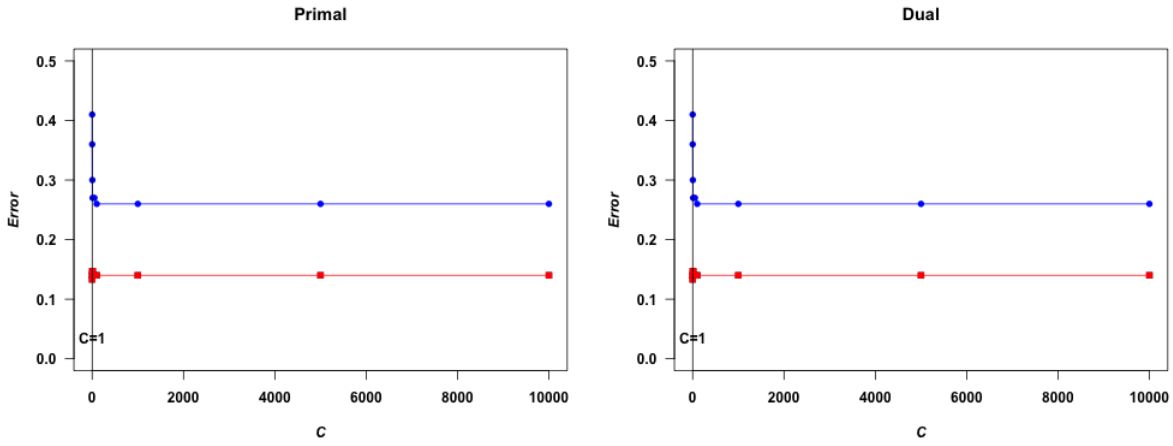

Figure 2: Error of primal and dual model according to C in training set (red line) and test set (blue).

Table 4: Hyperplanes according to C.

		Primal			Dual		
	C	β_0	β_1	β_2	β_0	β_1	β_2
1	0.5	-3.9742	2.9508	2.8103	-3.9742	2.9508	2.8103
2	1	-4.5811	3.5374	3.3527	-4.5811	3.5374	3.3527
3	5	-6.1031	5.3812	4.0359	-6.1031	5.3812	4.0359
4	10	-6.7204	6.2673	4.1475	-6.7204	6.2673	4.1475
5	50	-7.2	6.8571	4.2857	-7.2	6.8571	4.2857
6	100	-7.3429	7.102	4.2857	-7.3429	7.102	4.2857
7	1000	-7.4115	7.1605	4.321	-7.4115	7.1605	4.321
8	5000	-7.4115	7.1605	4.321	-7.4115	7.1605	4.321
9	10000	-7.4115	7.1605	4.321	-7.4115	7.1605	4.321

and also the most important analysis is that dataset is not *so* separable[2].

On the other hand, we just have some clues related to the accomplishment of strong duality due to obtained errors but get the variables results must be important to support this claim. Such as Table 4 we have the same variables' results, so we can claim that the strong duality is verified in this case.

Now, using Python library scikit-learn we have similar results according to the same dataset but with a considerable reduction of running time. Also, as we already know the optimal solution for each proposed parameter setting (C) we will just show the best solution of that one obtained from this process with its whole running time. Thus, the obtained hyperplane for primal and dual formulation are $-0.3571+0.7449x_1+0.3977x_2$ and $-0.7582+1.5265x_1+0.8936x_2$, respectively. Also, their error are 17.67% in both formulations and their best tuning parameter C is equal to 0.5 in both cases as well. Finally, if we compare those results with those obtained from the optimal solution we have that for primal formulation with we get the same conclusion related to C but with an error of 6.13% in both cases (remind that the strong duality is verified), but if we compare the dual formulation where we have discrepancies results do not change a lot, i.e. the error of optimal solution is 6.27%. Anyway, running times decrease from 2.0749 to 0.0517 in primal formulation and from 8.0472 to 0.2675.

[2]Note that despite of we are not able to plot the data (4D) we can use the definition of $C \to \infty$ when data is completely separable, i.e. we can get a hyperplane that separates with 100% of accuracy the data considering their classes.

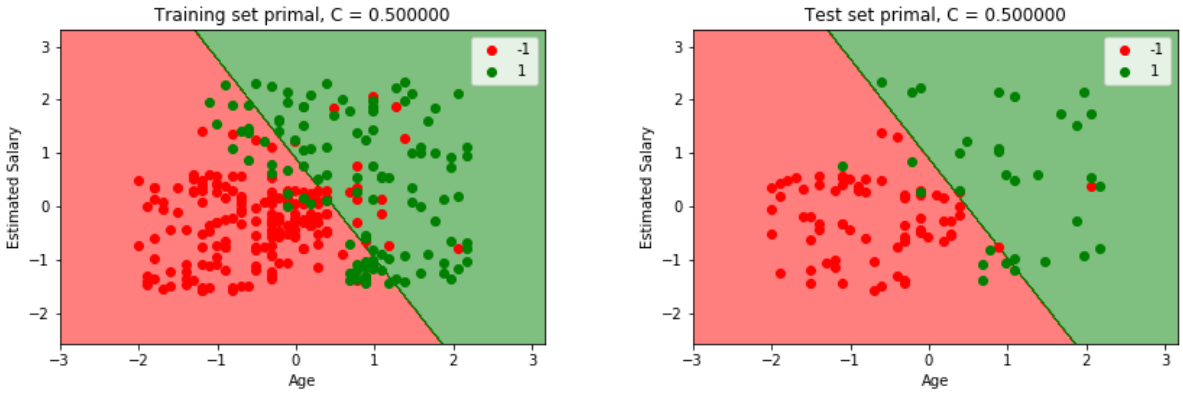

Figure 3: Classification of observations in training set and test set according to best parameters setting for primal formulation.

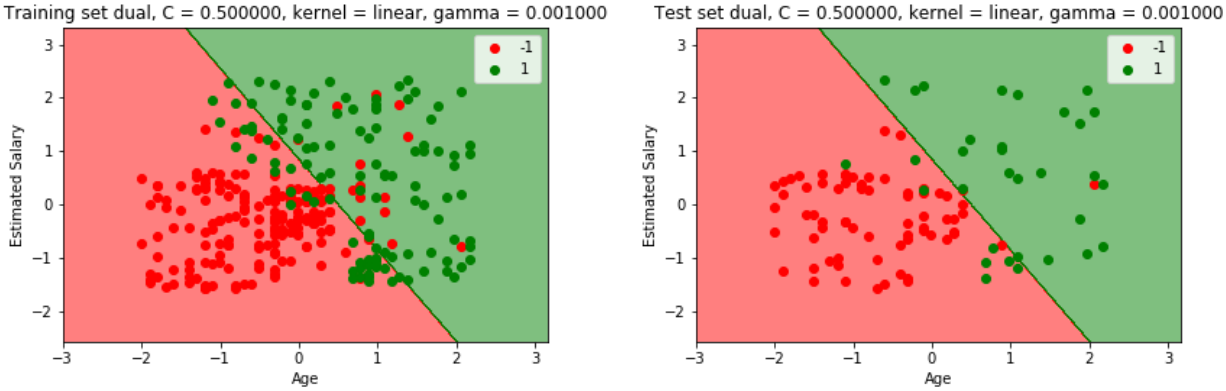

Figure 4: Classification of observations in training set and test set according to best parameters setting for dual formulation.

5. Conclusion

As we can show throughout this work, it is possible the get a hyperplane that separates the data according to two possible given classifications (a.k.a. supervised learning) including a parameter that allow us to incorporate some flexibility to work with that data which could be considered as non-separable. On the other hand, despite of we worked with not much observations (1000 and 400, respectively) running times are completely feasible to run it in a desktop computer, also we can highlight that the solution obtained from an algorithm (using scikit-learn) is quite close to the optimal solution which means that we would replicate it for a dataset with thousands of observations and get a closer solution to that provided for a solver, considering that the solver running time is much more high than the algorithm. Finally, according to obtained results it must be important to recognize how much separable is our data because of this provide us an idea of the order of parameter C we should use and which kind of formulation to use given that an approach to a non-separable dataset could be done applying a kernel different that the linear kernel shown here.

For further works we propose to use a k-fold-cross validation process to avoid a possible phenomena of over fitting and to use the random selection of observations to calibrate our model and compare our results against another classification models like logistic regression, neural networks, random forest or linear discriminant analysis (LDA).

References

[1] *The elements of statistical learning*, chapter 12. Springer, 2017.

[2] A. Ben-Hur, D. Horn, H. T. Siegelmann, and V. Vapnik. Support vector clustering. *Journal of machine learning research*, 2(Dec):125–137, 2001.

[3] C. M. Bishop. *Pattern recognition and machine learning*. Springer Science+ Business Media, 2006.

[4] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[5] P. Danenas and G. Garsva. Selection of support vector machines based classifiers for credit risk domain. *Expert Systems with Applications*, 42(6):3194–3204, 2015.

[6] H.-J. Huppertz, L. Möller, M. Südmeyer, R. Hilker, E. Hattingen, K. Egger, F. Amtage, G. Respondek, M. Stamelou, A. Schnitzler, et al. Differentiation of neurodegenerative parkinsonian syndromes by volumetric magnetic resonance imaging analysis and support vector machine classification. *Movement Disorders*, 31(10):1506–1517, 2016.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[8] M. D. Sacchet, G. Prasad, L. C. Foland-Ross, P. M. Thompson, and I. H. Gotlib. Support vector machine classification of major depressive disorder using diffusion-weighted neuroimaging and graph theory. *Frontiers in psychiatry*, 6:21, 2015.

[9] B. Scholkopf and A. J. Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.

[10] A. J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, 2004.

[11] Z. Taha, R. M. Musa, A. P. A. Majeed, M. M. Alim, and M. R. Abdullah. The identification of high potential archers based on fitness and motor ability variables: A support vector machine approach. *Human movement science*, 57:184–193, 2018.

[12] J. West and M. Bhattacharya. Intelligent financial fraud detection: a comprehensive review. *Computers & security*, 57:47–66, 2016.

[13] P. Wolfe. A duality theorem for non-linear programming. *Quarterly of applied mathematics*, 19(3):239–244, 1961.