DARIA KUDOIAR
dariakudoiar@gmail.com

Introduction:

A Support Vector Machine is a supervised machine learning, SVM classification algorithm does extremely complex data transformations to separate the data based on the labels or outputs defined. It follows a technique called the kernel trick to transform the data and based on these transformations, it finds an optimal boundary between the possible outputs.

Question 1. SVM classification for Simulated Data

STEP 1

We generate numbers using random sampling of uniform distribution over the interval [-2; 2]. We choose 16 random numbers for matrix $A_{ij}$with $i = 1, 2, 3, 4$ and $j = 1, 2, 3, 4$. Also, we select 4 random numbers B_i with $i = 1, 2, 3, 4$ and select 1 random number c. We have fixed A, B, c as below.

A: $\begin{bmatrix} 0.2 & 0.83 & -0.84 & 0.04 \\ 1.57 & 1.59 & -1.49 & -1.17 \\ -1.79 & -0.24 & -1.88 & -0.17 \\ 0.59 & -0.89 & 0.71 & 0.36 \end{bmatrix}$

B: $[-1.9 \quad 0.24 \quad -0.96 \quad -0.33]$

c: -0.87

We define the separator polynomial of degree 2 in the 4 variables x_i where $i = 1, 2, 3, 4$:

$$Pol(x) = \sum_i \sum_j A_{ij} x_i x_j + \sum_i B_i x_i + \frac{c}{20}$$

DARIA KUDOIAR
dariakudoiar@gmail.com

STEP 2

We generate numbers using random sampling of uniform distribution over the interval [-2; 2]. Select 10000 vectors $x^1 \ldots x^{10000}\ in\ R^4$. For each selected x^n we calculate $Pol(x^n)$ and store it as a variable $U(n)$, define sign of $U(n)$ and store it as the variable $y(n)$.

We define two target classes $CL(+1)$ and $CL(-1)$, where CL(1) is a set of all x^n such that $y(n) = +1$ and $CL(-1)$ is a set of all x^n such that $y(n) = -1$.

We randomly choose a sample of 5000 cases out of original10000 cases (2500 cases for $CL(+1)$ and 2500 cases for $CL(-1)$). Next step is center and rescale data set of 5000 cases such that standardized data set has $Mean = 0\ and\ SD = 1$. We split data set of 5000 cases into Training set and Test set by using proportions 80% and 20%, respectively. Also define a true response for each case and divide the data set into two data frames Data and Target. We make sure that $CL(+1)$ and $CL(-1)$ represent in Training and Test set accordingly.

We have a shape of Training set defined as ‘x_train’ equals to 4000 cases and 4 columns, Test set defined as ‘x_test’ equals to 1000 cases and 4 columns Respectively we choose Target set with shape 4000 rows and 1 column for Training set defined as ‘y_train’, Target set with shape 1000 rows and 1 column for Test set defined as ‘y_test’.

DARIA KUDOIAR
dariakudoiar@gmail.com

Question 2. SVM classification by Linear Kernel

The most basic way to use a Support Vector classification is with a Linear Kernel, which means the decision boundary is a straight line (or hyperplane in higher dimensions). To train our model and implement Support Vector Classification we use Scikit-Learn Python library.

Goal: identify the optimal separating hyperplane which divide the data into different planes so that it can find a best possible grouping of different classes, in our case $CL(+1)$ and $CL(-1)$ and predict the class of the unseen data. Maximizing the distance between the nearest points of each class and the hyperplane would result in an optimal separating hyperplane. This distance is called the margin. The optimal hyperplane is the one which has the biggest margin.

Kernel parameters selects the type of hyperplane used to separate the data. First we train our model using Linear Kernel. To train the model we fix arbitrarily parameter Cost = 5.

Run the SVM() on Train set we get as an output list of support vectors. We compute the number of support vectors and the ratio of support vectors as

$$S = \frac{Number\ of\ support\ vectors}{Length\ of\ training\ set}$$

In our specific case Length of training set = 4000.
We get results:

Number S of support vectors for Linear Kernel	3149
Ratio of support vectors for Linear Kernel	0.79

Percentages of correct prediction for Training set is 64%. We train our model on Train set and its ready to do classification task on unseen data – Test set. We run our model on Test set and get the percentages of correct prediction for Test set is 61%.

We compute confusion matrices for Training and Test set and converted it in terms of frequencies:

DARIA KUDOIAR
dariakudoiar@gmail.com

Table 1: Converted confusion matrix for Training data set

Predicted class / True class	CL(– 1)	CL(1)
CL(– 1)	66%	34%
CL(1)	37%	63%

Table 2: Converted confusion matrix for Test data set

Predicted class / True class	CL(– 1)	CL(1)
CL(– 1)	65%	35%
CL(1)	43%	57%

A confusion matrix is a summary of prediction results on a classification problem.

Rows represent True class and columns represent Predicted class. We can say by analyzing confusion matrix for Train set that it was correctly predicted 66% of CL(-1) and 63% of CL(1) and classification model predict 34% of CL(-1) as CL(1), and 37% of CL(1) as CL(-1)

We analyze confusion matrix for Test set and it was correctly predicted 65% of CL(-1) and 57% of CL(1) and classification model predict 35% of CL(-1) as CL(1), and 43% of CL(1) as CL(-1).

We calculate confidence intervals on the performance of the model to provide a calibrated and robust indication of model's skill.

DARIA KUDOIAR
dariakudoiar@gmail.com

Table 3: Classification Accuracy and Classification Errors, Confidence Intervals for CA and CE

	Percentage of Accuracy	95% Confidence Interval for Accuracy	Percentage of Classification Errors	95% Confidence Interval for Classification Errors
Training Set	64%	[62.87;65.83]	36%	[32.68; 38.62]
Test Set	61%	[57.98; 64.02]	39%	[35.98;42.02]

By comparing performances on Training and Test sets we see that Performance on Train set is bigger than on Test set by 3%, which is expected to be, because model supposed to perform better on known data and have less percentage of accuracy on unseen data.

There is a 95% likelihood that the confidence interval [62.87; 65.83] covers the true accuracy of the model on familiar data and interval [57.98; 64.02] covers the true accuracy of the model on unseen data. Furthermore, we are 95% confident that the true interval [35.98; 42.02] covers the true classification error of the model on unseen data. However, we see that confidence intervals overlap and to be more confident in the results we would like to se that intervals do not overlap.

Also, we can say that overall percentage of correct predictions is quite small. But it is clear that no linear discrimination could be able to separate our data, since we have 5000 vectors $x^1 \dots x^{5000}\ in\ R^4$.

We may see better performance of the model after tune parameters and use different basis Kernels such as Radial and Polynomial.

Question 3. Optimization of the parameter 'C'

'C' is the regularization parameter of the error term. It controls the tradeoff between smooth decision boundary and classifying the training points correctly. The function for tuning the parameters available in Scikit-Learn is called GridSearchCV(). It found the best parameter by running the model N times with fixed cross validation value (cv = 10).

DARIA KUDOIAR
dariakudoiar@gmail.com

We select a list of 6 values for the 'Cost' parameter. C = [2, 5, 10, 25, 30, 50]. As a result, we get that the best value for the parameter 'Cost' is equal to 10.

Now, we run the model for SVM classification by Linear Kernel with the best parameter 'Cost' = 10 and evaluate the performance.

SVM classification by Linear Kernel with optimized parameter 'Cost'

Training set:

Number S of support vectors for linear Kernel	3150
Ratio of support vectors for linear Kernel	0.79

Percentages of correct prediction for Training set is 64.42% and on Test set is 61%. We compute confusion matrices for Training and Test set and converted it in terms of frequencies:

Table 4: Converted confusion matrix for Training data set

Predicted class / True class	CL(– 1)	CL(1)
CL(– 1)	66%	34%
CL(1)	37%	63%

From confusion matrix for Train set we can say that 66% of CL(-1) and 63% of CL(1) were correctly predicted, however algorithm predicts 34% of CL(-1) as CL(1), and 37% of CL(1) as CL(-1)

Table 5: Converted confusion matrix for Test data set

Predicted class / True class	CL(– 1)	CL(1)
CL(– 1)	65%	35%

DARIA KUDOIAR
dariakudoiar@gmail.com

CL(1)	43%	57%

By analyzing confusion matrix for Test set we see that it was correctly predicted 65% of CL(-1) and 57% of CL(1) and classification model predict 35% of CL(-1) as CL(1), and 43% of CL(1) as CL(-1).

Table 6: Classification Accuracy and Classification Errors, Confidence Intervals for CA and CE

	Percentage of Accuracy	95% Confidence Interval for Accuracy	Percentage of Classification Errors	95% Confidence Interval for Classification Errors
Training Set	64.42%	[62.94; 65.9]	35.58%	[32.61; 38.55]
Test Set	61%	[57.98; 64.02]	39%	[35.98;42.02]

By comparing performances on Training and Test sets with 'Cost' = 5 and the best 'Cost' = 10, we see that Performance on Train set become slightly bigger (64.42%) with the same ratio of support vectors. However, percentage of correct predictions on Test set remains the same as before which is 61%. In addition, error percentage are quite high for both Train and Test set.

We can assume that the decision boundary seems to be in a good place and hyperplane already has close to the lowest possible misclassification error and increasing Cost would no longer have an effect. Also, performances on Test and Train set is low but we expect that, since we have nonlinear separable data and we try to separate it by Linear Kernel.

To get better accuracy we can try to find value for the Cost parameter out of much smaller values, to minimize the support values instead of misclassification error, but in this case probably we can get even lower result for accuracy.

DARIA KUDOIAR
dariakudoiar@gmail.com

Comparison Pol(x) vs SVM(x)

To compare the separator, we had initially defined to the separator that was constructed by SVM we plot a graph of Pol(x) vs SVM(x). Where Pol(x) is computed values by formula $Pol(x) = \sum_i \sum_j A_{ij} x_i x_j + \sum_i B_i x_i + \frac{c}{20}$

and decision values of SVM(x), which we obtained by method model.decision_function() that Scikit_Learn Python library has. We are interested to compare a sign and how far are the observations from each other.

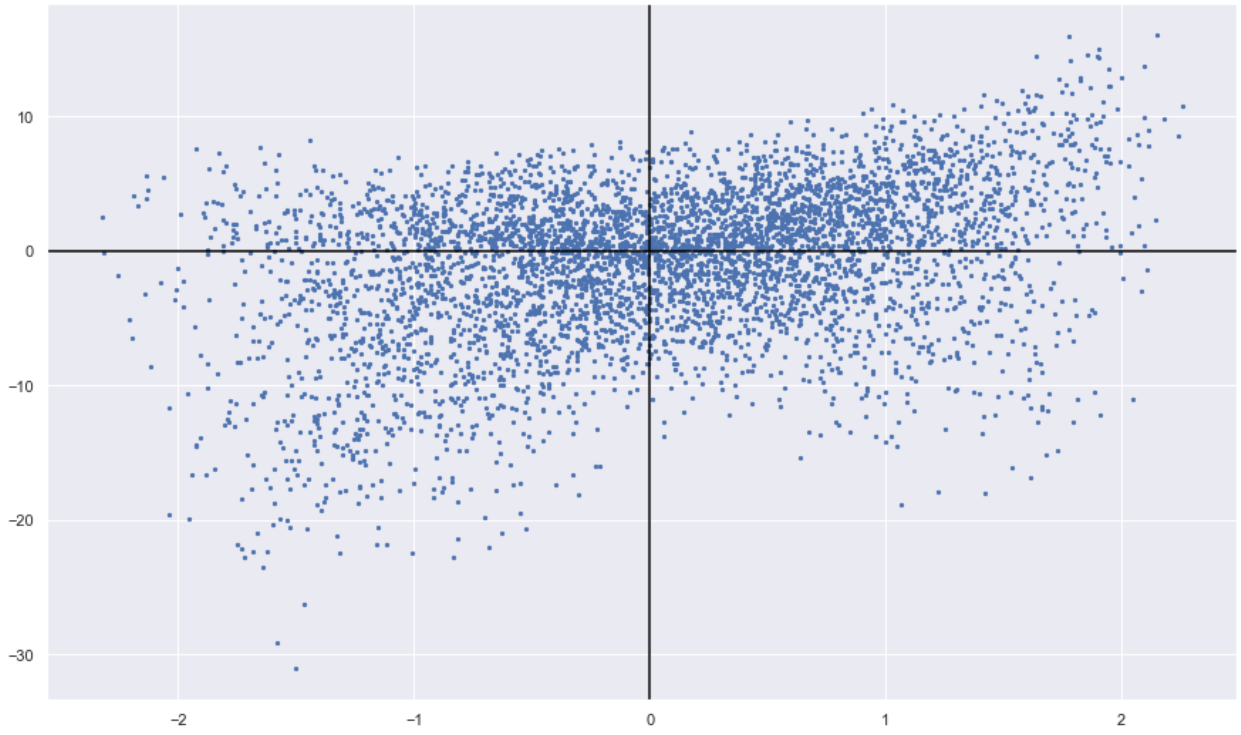

Figure 1: Comparison of separators Pol(x) vs SVM(x) for Linear Kernel

We plot values that we calculated for Pol(x) on x-axis VS decision values for SVM with best parameters for Linear Kernel on y-axis. Pattern of scatterplot needs to be located in III and I quadrants, we want to see that for each CL1 classified by Pol(x) we have matched CL1 classified by SVM(x). However, from the plot, we see that observations are highly spread out and there is approximately 40% of misclassifications, since a high amount of data points located at II and VI quadrants, which means that positive/negative output sign of Pol(x) there were negative/positive sign of decision values for SVM(x).

DARIA KUDOIAR
dariakudoiar@gmail.com

Question 4. SVM classification by radial kernel

Gaussian RBF (Radial Basis Function) is a popular Kernel method used in SVM models. RBF kernel is a function whose value depends on the distance from the origin or from some point. Gaussian Kernel is of the following format:

$$K(x, y) = e^{-\gamma\|x-y\|^2}$$

Hyperplane 'RBF' uses a nonlinear hyper-plane to separate the data and in this case, we expect to have high accuracy of classification task for our data set.

We select the 'Cost' parameter C = 10 which was defined as the best for Linear Support Vector Classifier and fix arbitrarily the parameter 'Gamma' = 1.

'Gamma' is a parameter of the RBF kernel and can be thought of as the 'spread' of the kernel and therefore the decision region. When 'Gamma' is low, the 'curve' of the decision boundary is very low and thus the decision region is very broad. When gamma is high, the 'curve' of the decision boundary is high, which creates islands of decision-boundaries around data points.

We get results below.

Number S of support vectors for Radial Kernel	427
Ratio of support vectors for Radial Kernel	0.11

Percentages of correct prediction for Training set is 99.62% and correct prediction for Test set is 96.9%, which is ~3% less than on Train set.

Table 7: Converted confusion matrix for Training data set

Predicted class / True class	CL(– 1)	CL(1)
CL(– 1)	99.95%	0.05%
CL(1)	0.07%	99.3%

Confusion matrix shows a quite high accuracy on Train set, Cl(-1) and CL(1) were classified correctly in 99.95% and 99.3% cases, respectively. In addition,

misclassification percentage very small, which is less then 1% for both classes CL(-1) and CL(1).

Table 8: Converted confusion matrix for Test data set

Predicted class / True class	CL(– 1)	CL(1)
CL(– 1)	96.96%	3.04%
CL(1)	3.16%	96.84%

For Train set we have a correct prediction of CL(-1) as 96.96% and 96.84% for CL(1), which is slightly less numbers compared to the Train performance.

Table 9: Classification Accuracy and Classification Errors, Confidence Intervals for CA and CE

	Percentage of Accuracy	95% Confidence Interval for Accuracy	Percentage of Classification Errors	95% Confidence Interval for Classification Errors
Training Set	99.62%	[99.43; 99.81%]	0.4%	[0.1; 0.76]
Test Set	96.9%	[95.83; 97.97%]	3.1%	[2.03; 4.17]

Accuracy on both Training and Test sets is quite high, which is 99.62% and 96.9%, respectively. However, we see situation when confidence intervals for accuracy of Train and Test sets do not overlap each other, which means that we very confident that the true percentages of accuracy lie in the intervals. We expect to see even higher performance of the model after tune parameters.

DARIA KUDOIAR
dariakudoiar@gmail.com

Question 5. Optimization of the parameters "Cost" and "Gamma"

We select 5 values for the "Cost " parameter C = [5, 10, 15, 30, 50] and 5 values for the parameter "Gamma" Gamma = [0.01, 0.1, 0.25, 0.5, 1]. We ran multiple times tune function and may assume that increasing gamma leads to overfitting as the classifier tries to perfectly fit the training data. We get the best pair of parameters as C = 30 and Gamma = 0.25.

SVM classification by Radial Kernel with optimized parameter ‘Cost” and "Gamma"

Training set:

Number S of support vectors for Radial Kernel	272
Ratio of support vectors for Radial Kernel	0.07

Percentages of correct prediction for Training set is 99.7% with numbers of support vectors decreased from 427 to 272 (and a ratio decreased from 0.11 to 0.07). and accuracy on Test set is 97.4%.

We compute confusion matrices for Training and Test set and converted it in terms of frequencies:

Table 10: Converted confusion matrix for Training data set

Predicted class / True class	CL(– 1)	CL(1)
CL(– 1)	99.6%	0.4%
CL(1)	2.5%	99.75%

We can say from confusion matrix for Train set that it was correctly predicted 99.6% of CL(-1) and 99.75% of CL(1) and classification model predict 0.4% of CL(-1) as CL(1), and 0.25% of CL(1) as CL(-1)

DARIA KUDOIAR
dariakudoiar@gmail.com

Table 11: Converted confusion matrix for Test data set

Predicted class / True class	CL(– 1)	CL(1)
CL(– 1)	99.4%	0.6%
CL(1)	2.6	97.4%

For Test set that it was correctly predicted 99.4% of CL(-1) and 97.4% of CL(1) and classification model predict 0.6% of CL(-1) as CL(1), and 2.6% of CL(1) as CL(-1).

Table 12: Classification Accuracy and Classification Errors, Confidence Intervals for CA and CE

	Percentage of Accuracy	95% Confidence Interval for Accuracy	Percentage of Classification Errors	95% Confidence Interval for Classification Errors
Training Set	99.7%	[99.8; 99.5]	0.3%	[0.67; 0.03]
Test Set	99.4%	[99.7; 98.7]	0.6%	[1.08; 0.12]

By comparing performances on Training and Test sets with the best parameters, as we expected, we get higher percentage of accuracy and smaller number, ratio of support vectors with optimized parameters for Radial Kernel, number of vectors reduced by 2. Performances on both Train and Test set seem to be quite close to 100%. SVM classification has done a very good of classification task on our data set.

DARIA KUDOIAR
dariakudoiar@gmail.com

Comparison Pol(x) vs SVM(x)

We plot values that we calculated for Pol(x) on x-axis VS decision values for SVM with best parameters for Radial Kernel on y-axis.

We see that they match in overall, means that for each actual class 1 or -1 there were defined correct prediction of class. However, we can see that data points are highly scattered in the intervals somewhere (-30: -10) and (10; 30). We would like to see “an ideal” situation where all points lie along line.

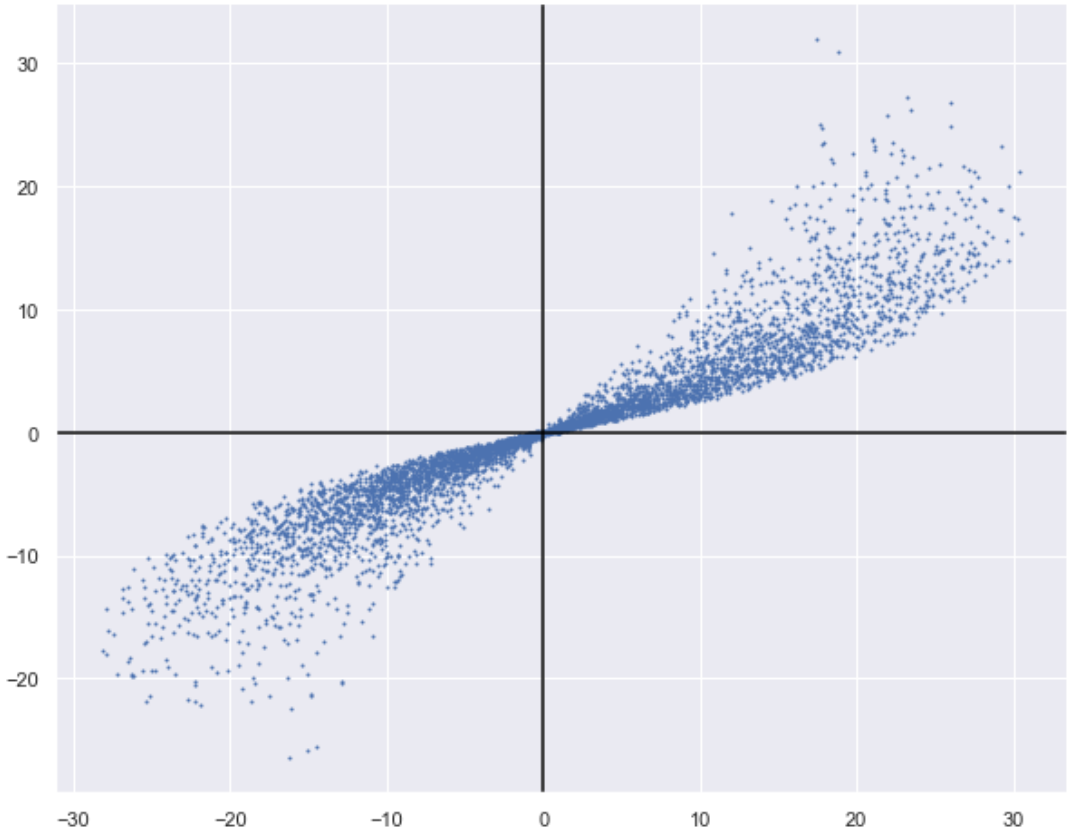

Figure 2: Comparison of separators Pol(x) vs SVM(x) for Linear Kernel

DARIA KUDOIAR
dariakudoiar@gmail.com

Question 6. SVM classification by Polynomial Kernel

In general, the polynomial kernel is defined as:

$$K(x, y) = (a + \langle x, y \rangle)^r$$

where a – constant term, r – degree of polynomial. In the polynomial kernel, we calculate the dot product by increasing the power of the kernel.
In our case we use Polynomial Kernel of degree 4 for SVM classification. We expect the best performance for this model, because initially separator was defined as Polynomial of degree 2.

We obtain the following results:

Number S of support vectors for Polynomial Kernel	168
Ratio of support vectors for Polynomial Kernel	0.04

Percentages of correct prediction for Training set is 99.72% and Test set is 99.5%

We compute confusion matrices for Training and Test set and converted it in terms of frequencies:

Table 13: Converted confusion matrix for Training data set

Predicted class / True class	CL(– 1)	CL(1)
CL(– 1)	99.7%	0.3%
CL(1)	0.25%	99.75%

Table 14: Converted confusion matrix for Test data set

Predicted class / True class	CL(– 1)	CL(1)
CL(– 1)	99.2%	0.8%
CL(1)	0.2%	99.8%

DARIA KUDOIAR
dariakudoiar@gmail.com

Table 15: Classification Accuracy and Classification Errors, Confidence Intervals for CA and CE

	Percentage of Accuracy	95% Confidence Interval for Accuracy	Percentage of Classification Errors	95% Confidence Interval for Classification Errors
Training Set	99.72%	[99.88%; 99.56%]	0.28%	[0.61; 0.05]
Test Set	99.5%	[99.6%; 99.36%]	0.5%	[0.94; 0.06]

Accuracy on both Training and Test sets is very close to 100%. Performance on test set is slightly less than Performance on Train set which is 99.72% and 99.5%, respectively. High performance on the SVM classification with Polynomial basis Kernel ma means that it works very well with our data and did almost excellent job for classification task.

Optimization of the two parameters "a" and ''cost"

We select 5 values for the "Cost " parameter C = [5, 7, 10, 15, 30] and 5 values for the parameter "a" a = [1, 5, 10, 15, 20]. We get the best pair of parameters as C = 30 and a = 20

SVM classification by Polynomial Kernel with optimized parameter “Cost” and "a"

We obtain the following results:

Number S of support vectors for Polynomial Kernel	67
Ratio of support vectors for Polynomial Kernel	0.017

Percentages of correct prediction for Training set is 99.8% and Test set is 99.6%

Table 16: Converted confusion matrix for Training data set

DARIA KUDOIAR
dariakudoiar@gmail.com

Predicted class / True class	CL(– 1)	CL(1)
CL(– 1)	99.8%	0.2%
CL(1)	0.25%	99.75%

99.8% of CL(-1) and 99.75% of CL(1) were predicted correctly by SVM algorithm, while model mistakenly predict 0.2% of CL(-1) as CL(1), and 0.25% of CL(1) as CL(-1).

Table 17: Converted confusion matrix for Test data set

Predicted class / True class	CL(– 1)	CL(1)
CL(– 1)	99.4%	0.6%
CL(1)	0.2	99.8%

We have quite similar situation for accuracy on Test set compared to Train set. It was correctly predicted 99.4% of CL(-1) and 99.8% of CL(1) and classification model predict 0.6% of CL(-1) as CL(1), and 0.2% of CL(1) as CL(-1)

Table 18: Classification Accuracy and Classification Errors, Confidence Intervals for CA and CE

	Percentage of Accuracy	95% Confidence Interval for Accuracy	Percentage of Classification Errors	95% Confidence Interval for Classification Errors
Training Set	99.8%	[99.91%; 99.6%]	0.2%	[0.6; 0.04]
Test Set	99.6%	[99.7%; 99.02%]	0.4%	[0.9; 0.03]

Taking into account confidence intervals for both Train and Test sets, we can conclude that accuracy is extremely close to 100%. We see that the model with the

best parameters have higher accuracy with a smaller number of support vectors, numbers of support vectors decreased by 2.5. Still performance on Train set slightly bigger than on Test set but comparing confidence intervals of Accuracy for both Train and Test set and classification errors we can conclude that our model did excellent job on classification task.

Comparison Pol(x) vs SVM(x)

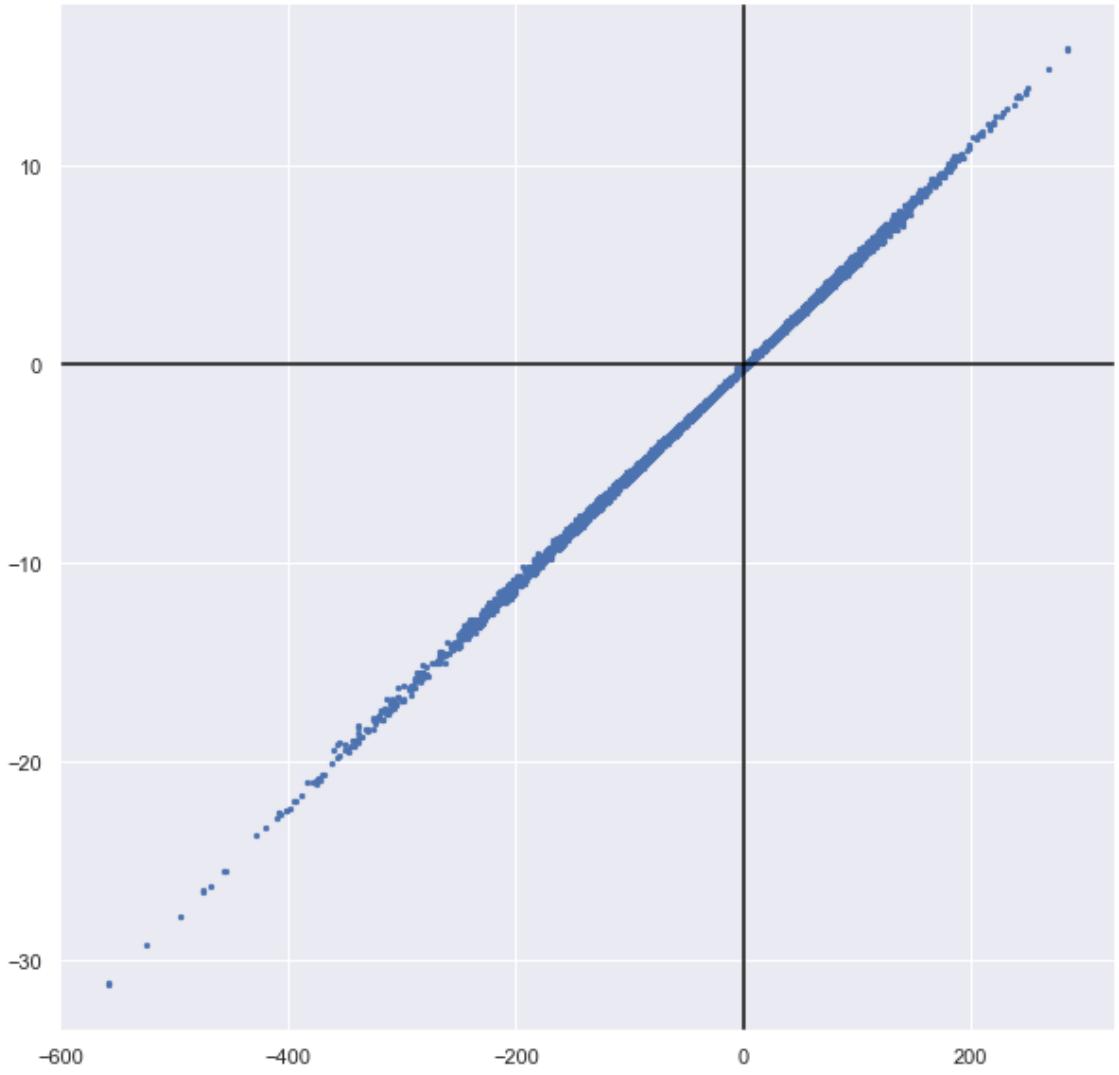

Figure 3: Pol(x) vs SVM(x) for Polynomial Kernel

We plot values that we calculated for Pol(x) on x-axis VS decision values for SVM with best parameters for Polynomial Kernel on y-axis. We see that they perfectly

match, means that for each actual class 1 or -1 there were defined correct prediction of class. We see data points lie along line, and we don't have data points in "incorrectly" classified II and IV quadrants, which means there is no or very small number of misclassification errors.

Conclusion:

We ran SVM classification for Simulated Data with different basis type of Kernels: Linear Kernel, Radial Kernel and Polynomial Kernel. We can conclude that Radial and Polynomial Kernels work very well for our simulated data. We ran Linear model basically to see that it`s not working well, since data were nonlinear separable and there is no linear discrimination could be able to separate nonlinear separable data.

By comparing performances on Train and Test sets with best parameters for Radial and Polynomial Kernels we can say that both did excellent job for classification task on our data. However, Polynomial Kernel has slightly bigger percentage of accuracy with smaller ratio of support vectors less than 2% compared to Radial Kernel. Analyzing scatter plot Pol(x) vs SVM(x) we can conclude that the best Kernel for SVM classification for our data is Polynomial basis Kernel with accuracy 99.6% on unseen data and we are 95% confident that the true interval [99.02%; 99.7%] covers the true accuracy of classification.