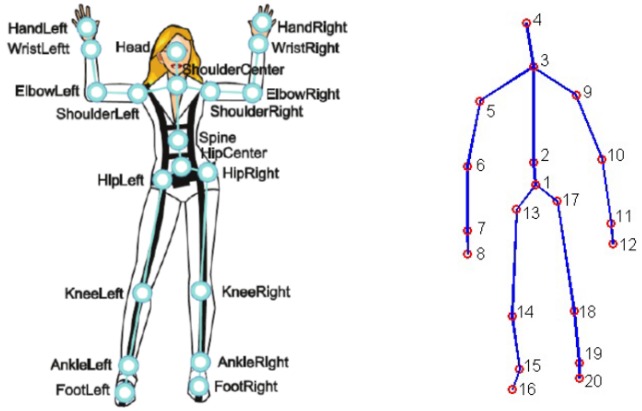
Project 3 Deliverable 2

John Ripple
CSCI
Colorado School of Mines
Golden, United States
jripple@mines.edu

***Abstract*—This document discusses using cleaned human skeleton data formatted into histograms with RAD and a custom algorithm to create an SVM to classify if human skeleton data is performing a certain action.**

I. Introduction

A 3D camera was used to capture Human Skeleton data which included 20 joint positions shown in figure 1. To interpret what movement a person is making, the raw skeleton data was featurized. The featurization was done by turning each joint's data into histograms through the RAD algorithm with different joints used.



(a) Names (b) Indices

Fig. 1: Skeleton joint names and indices from Kinect SDK.

II. RAD Algorithm

The RAD Algorithm works by calculating the 3D distance from the hip center joint to each of the extremity joints. RAD also calculates the 3D angles between the vectors created from the hip center joint to each extremity. Both features are shown in figure 2. For every frame, both distance and angle are calculated and stored. The distance values for a joint for the entire file are aggregated and turned into a histogram. The angle values for a joint calculated from a file are also aggregated and turned into a histogram.

III. Custom Algorithm

The secondary custom algorithm used for featurization was RAD but instead of measuring distances and angles to the end of extremities, the intermediate joints were used (elbows, knees, and neck). Originally, HJPD was the secondary algorithm used but the SVM accuracy was 16%, and no matter what parameters were used with the SVM or when featurizing the data, the accuracy would not improve. The RAD adjacent algorithm was chosen as a replacement.

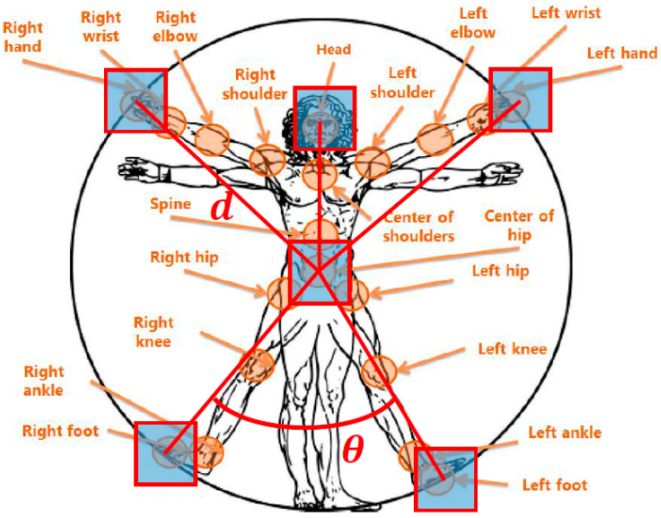


Fig. 2: Illustration of human representation based on relative distance and angles of star skeleton.

IV. Histogram Creation

For either algorithm used, feature data is generated and turned into a histogram. To turn 2D list feature data (table I) of joint data per frame into histograms, first, the data was transposed (table II). Transposing the data takes the data from every joint per frame into all data for a joint in a single list/row.

TABLE I:
Featurization data for joints per frame

Frame 1:	Joint 1 Data	Joint 2 Data	...
Frame 2:	Joint 1 Data	Joint 2 Data	...
Frame 3:	Joint 1 Data	Joint 2 Data	...
...	...	...	...

Each row, representing a single joint's data, was then fed into numpy.histogram() with 20 bins. Every histogram has 20

TABLE II: Transposed featurization data

Frame 1:	Frame 2:	Frame 3:	...
Joint 1 Data	Joint 1 Data	Joint 1 Data	...
Joint 2 Data	Joint 2 Data	Joint 2 Data	...
...	...	...	...

bins, but the range for the histogram changes based on the data. The lower limit of the histogram is the minimum value found in the feature data (the row used in numpy.histogram()) while the upper limit is the maximum value found in the feature data. This allows every histogram to have the same number of bins while still allowing each histogram to not have large ranges with little to no data.

V. Support Vector Machine (SVM)

A support vector machine (SVM) is a supervised learning model that requires relatively little data because the model is dependent on support vectors or the closest points to the hyperplane. SVM works by creating a hyperplane that splits data categories as shown in figure 3. Features are then fed into the SVM and the SVM classifies what category the feature most likely belongs to allowing for recognition of data. In the case of this paper, the features used are histograms generated through RAD and the categories are what action the human skeleton data corresponds to.

The main ways to tweak SVM performance is by changing the kernel, C value, and gamma value. The kernel is a function that the SVM uses to classify categories of data. With this project, the types of kernels tested were linear, sigmoid, polynomial, or RBF. Through experimental testing, the RBF kernel was found to be the best for this application.

The C value controls the error or margins of an SVM. A smaller C value creates larger-margin hyperplanes (more error) meaning more misclassifications occur while training, while larger C values create smaller-margin hyperplanes (less error) and fewer misclassifications [2]. The C value must be experimentally found since having a very large C could lead to few misclassifications during training and many misclassifications during testing. This is because SVMs can be overfit to training data leading the SVM to only work on the training data.

Gamma is a hyper-parameter used with the RBF, polynomial, and sigmoid kernels. In general, gamma determines how much curvature is wanted at the decision boundary [3]. High gamma is more curvature (or fit) and low gamma is less curvature (less fit). What gamma to use is dependent on the data because too high of gamma can overfit the model just like a small C value can. Another way to think about it is that gamma controls the distance of influence of points.

VI. Results

A. RAD Algorithm SVM

The RAD algorithm SVM had the best result with and rbf kernel, C value of 1, and a gamma of 0.6 giving an accuracy of 64.58%. All other combinations of the parameters shown in Listing 1 are in Appendix A.

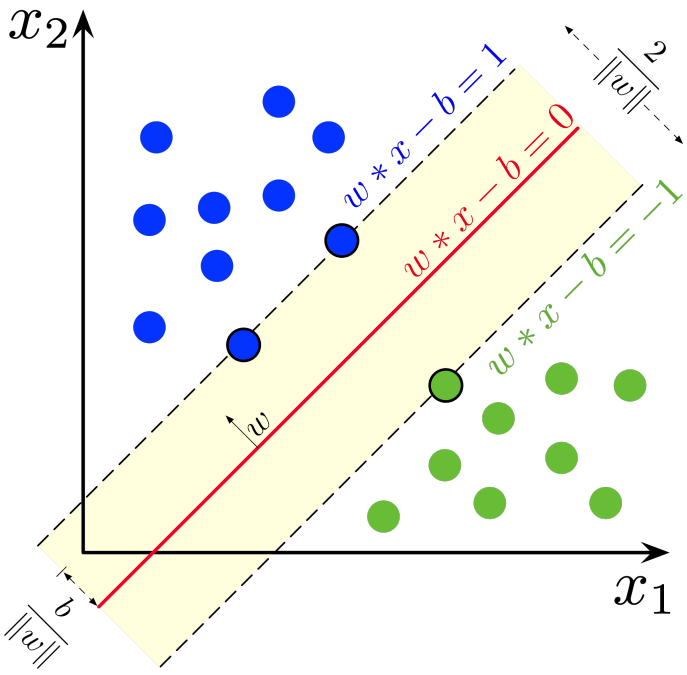


Fig. 3: Maximum-margin hyperplane and margins for an SVM trained with samples from two classes. Samples on the margin are called the support vectors [1].

Listing 1: Parameters used in creation of the SVM

```
parameters =
{'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000],
'kernel': ['rbf', 'linear', 'poly', 'sigmoid'],
'gamma': ['auto', 1, 0.1, 0.02, 0.001]}
```

Various combinations of C, gamma, and kernel functions were tried to increase the classification accuracy of the RAD featurization algorithm SVM model. The best kernel with default parameters was found to be RBF with an accuracy of 60.4% while linear was 56.25%, polynomial was 58.33% and sigmoid was 56.25%. Based on these initial findings RBF was chosen to have more testing done with C and gamma values.

The best C value found with the RBF kernel was anything below 1.0 all giving an accuracy of 62.5% while any C value above 1.0 gave an accuracy of $< 60.4\%$. Any value below 1.0 gave the same accuracy, even values very close to zero (1.0e-9). This means for this dataset much of the data is well grouped with few outliers to cause misclassifications, so punishing the model with a high C value does no good.

The best gamma value was 1.0 with an accuracy of 62.5%. Any gamma value smaller or larger than 1.0 gave accuracy $< 58.3\%$. For this dataset a gamma value of 1.0 allows the kernel to split the data effectively while not overfitting to the data. The worst model was caused by changing the gamma value and making it smaller than the default value. This happened because the gamma was too low, causing very little curvature at the boundary leading to the model being underfitted and not being able to train to the data very well.

Overall, the best model utilized an RBF kernel with a C value of 0.6 and gamma value of 1.0. All of the best SVMs tested used RBF instead of linear, polynomial, or sigmoid. This is most likely because the RBF has the most customizable based on the data with the gamma value. The RBF kernel can split categories and create support vectors better than the other kernels with this data. The best C and gamma values were found experimentally and strike the balance between overfitting the SVM to the data and being fit enough to classify features correctly. The larger C-value (above 0.6) SVMs suffered when predicting what action was chosen because it overfit to the training data. The same could be said for gamma values below 1.

B. Custom Algorithm SVM

For the custom featurization algorithm, the best SVM had an accuracy of 60.4%. Other metrics such as precision and the confusion matrix are listed in Appendix B. All the comparisons made for the RAD SVM hold for the modified custom SVM (modified RAD). The best kernel was RBF with all other kernels having comparable or poorer accuracy compared to RBF. Accuracy for every kernel without changing any other default parameters was RBF=54.2%, linear= 50%, polynomial=54.2%, sigmoid=43.8%. Based on these results, RBF was chosen over polynomial because further testing with the gamma parameter showed better results with the RBF kernel.

For the C parameter, anything below 1.4 had lower accuracy compared to anything above, corresponding to 58.5% below 1.4 and 60.4% above. This is what is expected to a point, lower C values create more misclassifications while high C values are more accurate. The unexpected thing is that very large C values (1.4 to 1000) don't seem to overfit to the data and cause the accuracy of the SVM to suffer. This most likely happens because C is a punishment for misclassifications, so over a certain threshold (1.4 in this case) the punishment is large enough that the model remains the same no matter what the punishment value is.

For the gamma parameter, the best results were found with a default gamma of 1 / (n_features * X.var()). This value comes out to 1.44. Anything above or below this gamma gets a worse accuracy (gamma=1.44 with an accuracy of 60.4%, gamma < 1.44 gets an accuracy of $< 50.0\%$, gamma > 1.44 gets an accuracy of $< 47.9\%$). This lines up with what is expected with gamma, there will be an experimentally derived value that maximizes the splitting effect of the kernel without overfitting the SVM to the data.

Based on all of the experiments, the best SVM for the custom featurization algorithm was an RBF kernel, a C value of 1.4, and a gamma of 1.44.

C. Algorithm Comparison

The two best featurization algorithms and SVMs had an accuracy of 64.58% and 60.42% corresponding to the RAD and Custom algorithms respectively. The difference between the two SVMs came down to the featurization algorithm, not the SVM. This is because the RAD algorithm uses the extremities (hands, head, feet) while the Custom algorithm is the same as RAD but uses the extremity joints (elbows, head, knees). By using the joints instead of the end of the extremities, a lot of data is lost, the movement of the joints closer to the body is not as expressive as at the end of the extremities. This loss of movement data makes it harder to classify features correctly.

Overall, both featurization algorithms produced poor SVMs that only classified human skeleton data correctly around 60% of the time. This is most likely due to the featurization algorithm used instead of the SVM tuning parameters. A better featurization method, such as HJPD or HOD might return better SVMs.

References

[1] Wikimedia Commons. File:svm margin.png — wikimedia commons, the free media repository, 2022. [Online; accessed 29-April-2023].

[2] Felipe. Choosing c hyperparameter for svm classifiers: Examples with scikit-learn, Aug 2019.

[3] A Man Kumar. C and Gamma in SVM, 2018. [Online; accessed 29-April-2023].

Appendix A
SVM Results for RAD Algorithm

Best SVM found (rbf): SVC(C=0.6, gamma=1)
Precision: [1. 0.66666667 0.5 0.75 1. 0.46666667]
Accuracy: 0.6458333333333334
Confusion Matrix: $\begin{bmatrix} 8 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 3 & 1 & 0 & 0 \\ 0 & 1 & 5 & 0 & 0 & 2 \\ 0 & 0 & 1 & 6 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 5 \\ 0 & 1 & 0 & 0 & 0 & 7 \end{bmatrix}$

SVC(C=0.001)
Precision: [1. 0.57142857 0.55555556 0.75 0.5 0.46153846]
Accuracy: 0.625
Confusion Matrix: $\begin{bmatrix} 7 & 1 & 0 & 0 & 0 & 0 \\ 0 & 4 & 3 & 1 & 0 & 0 \\ 0 & 0 & 5 & 0 & 1 & 2 \\ 0 & 0 & 1 & 6 & 0 & 1 \\ 0 & 1 & 0 & 1 & 2 & 4 \\ 0 & 1 & 0 & 0 & 1 & 6 \end{bmatrix}$

SVC(C=0.01)
Precision: [1. 0.57142857 0.55555556 0.75 0.5 0.46153846]
Accuracy: 0.625
Confusion Matrix: $\begin{bmatrix} 7 & 1 & 0 & 0 & 0 & 0 \\ 0 & 4 & 3 & 1 & 0 & 0 \\ 0 & 0 & 5 & 0 & 1 & 2 \\ 0 & 0 & 1 & 6 & 0 & 1 \\ 0 & 1 & 0 & 1 & 2 & 4 \\ 0 & 1 & 0 & 0 & 1 & 6 \end{bmatrix}$

SVC(C=0.1)
Precision: [1. 0.57142857 0.55555556 0.75 0.5 0.46153846]
Accuracy: 0.625
Confusion Matrix: $\begin{bmatrix} 7 & 1 & 0 & 0 & 0 & 0 \\ 0 & 4 & 3 & 1 & 0 & 0 \\ 0 & 0 & 5 & 0 & 1 & 2 \\ 0 & 0 & 1 & 6 & 0 & 1 \\ 0 & 1 & 0 & 1 & 2 & 4 \\ 0 & 1 & 0 & 0 & 1 & 6 \end{bmatrix}$

SVC(C=1)
Precision: [1. 0.5 0.5 0.85714286 0.57142857 0.36363636]
Accuracy: 0.6041666666666666
Confusion Matrix: $\begin{bmatrix} 7 & 1 & 0 & 0 & 0 & 0 \\ 0 & 5 & 1 & 1 & 0 & 1 \\ 0 & 1 & 3 & 0 & 1 & 3 \\ 0 & 0 & 1 & 6 & 0 & 1 \\ 0 & 1 & 1 & 0 & 4 & 2 \\ 0 & 2 & 0 & 0 & 2 & 4 \end{bmatrix}$

SVC(C=10)
Precision: [1. 0.42857143 1. 0.85714286 0.5 0.4]
Accuracy: 0.6041666666666666
Confusion Matrix: $\begin{bmatrix} 8 & 0 & 0 & 0 & 0 & 0 \\ 0 & 6 & 0 & 1 & 0 & 1 \\ 0 & 2 & 1 & 0 & 2 & 3 \\ 0 & 1 & 0 & 6 & 0 & 1 \\ 0 & 3 & 0 & 0 & 4 & 1 \\ 0 & 2 & 0 & 0 & 2 & 4 \end{bmatrix}$

SVC(C=100)
Precision: [1. 0.42857143 1. 0.85714286 0.5 0.4]
Accuracy: 0.6041666666666666
Confusion Matrix: $\begin{bmatrix} 8 & 0 & 0 & 0 & 0 & 0 \\ 0 & 6 & 0 & 1 & 0 & 1 \\ 0 & 2 & 1 & 0 & 2 & 3 \\ 0 & 1 & 0 & 6 & 0 & 1 \\ 0 & 3 & 0 & 0 & 4 & 1 \\ 0 & 2 & 0 & 0 & 2 & 4 \end{bmatrix}$

SVC(C=1000)
Precision: [1. 0.42857143 1. 0.85714286 0.5 0.4]
Accuracy: 0.6041666666666666
Confusion Matrix: $\begin{bmatrix} 8 & 0 & 0 & 0 & 0 & 0 \\ 0 & 6 & 0 & 1 & 0 & 1 \\ 0 & 2 & 1 & 0 & 2 & 3 \\ 0 & 1 & 0 & 6 & 0 & 1 \\ 0 & 3 & 0 & 0 & 4 & 1 \\ 0 & 2 & 0 & 0 & 2 & 4 \end{bmatrix}$

SVC()
Precision: [1. 0.5 0.5 0.85714286 0.57142857 0.36363636]
Accuracy: 0.6041666666666666
Confusion Matrix: $\begin{bmatrix} 7 & 1 & 0 & 0 & 0 & 0 \\ 0 & 5 & 1 & 1 & 0 & 1 \\ 0 & 1 & 3 & 0 & 1 & 3 \\ 0 & 0 & 1 & 6 & 0 & 1 \\ 0 & 1 & 1 & 0 & 4 & 2 \\ 0 & 2 & 0 & 0 & 2 & 4 \end{bmatrix}$

SVC(kernel='linear')
Precision: [0.77777778 0.25 0.8 0.63636364 0.5 0.44444444]
Accuracy: 0.5625

Confusion Matrix: $\begin{bmatrix} 7 & 1 & 0 & 0 & 0 & 0 \\ 2 & 2 & 1 & 3 & 0 & 0 \\ 0 & 0 & 4 & 0 & 1 & 3 \\ 0 & 0 & 0 & 7 & 0 & 1 \\ 0 & 4 & 0 & 0 & 3 & 1 \\ 0 & 1 & 0 & 1 & 2 & 4 \end{bmatrix}$

SVC(kernel='poly')
Precision: [0.8 0.4 0.625 0.54545455 0.66666667 0.45454545]
Accuracy: 0.5833333333333334
Confusion Matrix: $\begin{bmatrix} 8 & 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 1 & 3 & 0 & 0 \\ 0 & 0 & 5 & 0 & 0 & 3 \\ 0 & 0 & 2 & 6 & 0 & 0 \\ 0 & 1 & 0 & 2 & 2 & 3 \\ 0 & 2 & 0 & 0 & 1 & 5 \end{bmatrix}$

SVC(kernel='sigmoid')
Precision: [0.875 0.25 0.66666667 0.53846154 0.6 0.5]
Accuracy: 0.5625
Confusion Matrix: $\begin{bmatrix} 7 & 1 & 0 & 0 & 0 & 0 \\ 1 & 2 & 2 & 3 & 0 & 0 \\ 0 & 0 & 4 & 1 & 1 & 2 \\ 0 & 0 & 0 & 7 & 0 & 1 \\ 0 & 3 & 0 & 1 & 3 & 1 \\ 0 & 2 & 0 & 1 & 1 & 4 \end{bmatrix}$

SVC(gamma='auto')
Precision: [0.7 0.2 0.71428571 0.46666667 0. 0.45454545]
Accuracy: 0.5208333333333334
Confusion Matrix: $\begin{bmatrix} 7 & 1 & 0 & 0 & 0 & 0 \\ 2 & 1 & 1 & 4 & 0 & 0 \\ 0 & 0 & 5 & 1 & 0 & 2 \\ 0 & 0 & 0 & 7 & 0 & 1 \\ 1 & 1 & 1 & 2 & 0 & 3 \\ 0 & 2 & 0 & 1 & 0 & 5 \end{bmatrix}$

SVC(gamma=1)
Precision: [0.875 0.57142857 0.55555556 0.85714286 0.5 0.46153846]
Accuracy: 0.625
Confusion Matrix: $\begin{bmatrix} 7 & 1 & 0 & 0 & 0 & 0 \\ 1 & 4 & 2 & 1 & 0 & 0 \\ 0 & 0 & 5 & 0 & 1 & 2 \\ 0 & 0 & 1 & 6 & 0 & 1 \\ 0 & 1 & 1 & 0 & 2 & 4 \\ 0 & 1 & 0 & 0 & 1 & 6 \end{bmatrix}$

SVC(gamma=0.1)
Precision: [0.7 0.2 0.71428571 0.46666667 0. 0.45454545]
Accuracy: 0.5208333333333334
Confusion Matrix: $\begin{bmatrix} 7 & 1 & 0 & 0 & 0 & 0 \\ 2 & 1 & 1 & 4 & 0 & 0 \\ 0 & 0 & 5 & 1 & 0 & 2 \\ 0 & 0 & 0 & 7 & 0 & 1 \\ 1 & 1 & 1 & 2 & 0 & 3 \\ 0 & 2 & 0 & 1 & 0 & 5 \end{bmatrix}$

SVC(gamma=0.02)
Precision: [0.7 0.2 0.71428571 0.46666667 0. 0.45454545]
Accuracy: 0.5208333333333334
Confusion Matrix: $\begin{bmatrix} 7 & 1 & 0 & 0 & 0 & 0 \\ 2 & 1 & 1 & 4 & 0 & 0 \\ 0 & 0 & 5 & 1 & 0 & 2 \\ 0 & 0 & 0 & 7 & 0 & 1 \\ 1 & 1 & 1 & 2 & 0 & 3 \\ 0 & 2 & 0 & 1 & 0 & 5 \end{bmatrix}$

SVC(gamma=0.001)
Precision: [0.7 0.2 0.71428571 0.46666667 0. 0.45454545]
Accuracy: 0.5208333333333334
Confusion Matrix: $\begin{bmatrix} 7 & 1 & 0 & 0 & 0 & 0 \\ 2 & 1 & 1 & 4 & 0 & 0 \\ 0 & 0 & 5 & 1 & 0 & 2 \\ 0 & 0 & 0 & 7 & 0 & 1 \\ 1 & 1 & 1 & 2 & 0 & 3 \\ 0 & 2 & 0 & 1 & 0 & 5 \end{bmatrix}$

APPENDIX B
CUSTOM ALGORITHM SVM RESULTS

SVM results for Custom
Best SVM found (rbf): SVC(C=10.0, gamma=1.443)
Precision: [0.875 0.5 0.4 1. 0.5 0.44444444]
Accuracy: 0.6041666666666666
Confusion Matrix: $\begin{bmatrix} 7 & 0 & 0 & 0 & 1 & 0 \\ 1 & 4 & 1 & 0 & 1 & 1 \\ 0 & 2 & 2 & 0 & 2 & 2 \\ 0 & 0 & 1 & 6 & 0 & 1 \\ 0 & 1 & 0 & 0 & 6 & 1 \\ 0 & 1 & 1 & 0 & 2 & 4 \end{bmatrix}$

SVC(C=0.001)
Precision: [1. 0.375 0.375 0.85714286 0.4 0.42857143]
Accuracy: 0.5625
Confusion Matrix: $\begin{bmatrix} 8 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 3 & 0 & 1 & 1 \\ 0 & 0 & 3 & 1 & 3 & 1 \\ 0 & 1 & 1 & 6 & 0 & 0 \\ 0 & 2 & 0 & 0 & 4 & 2 \\ 0 & 2 & 1 & 0 & 2 & 3 \end{bmatrix}$

SVC(C=0.01)
Precision: [1. 0.375 0.375 0.85714286 0.4 0.42857143]
Accuracy: 0.5625

Confusion Matrix: $\begin{bmatrix} 8 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 3 & 0 & 1 & 1 \\ 0 & 0 & 3 & 1 & 3 & 1 \\ 0 & 1 & 1 & 6 & 0 & 0 \\ 0 & 2 & 0 & 0 & 4 & 2 \\ 0 & 2 & 1 & 0 & 2 & 3 \end{bmatrix}$

SVC(C=0.1)
Precision: [1. 0.375 0.375 0.85714286 0.4 0.42857143]
Accuracy: 0.5625
Confusion Matrix: $\begin{bmatrix} 8 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 3 & 0 & 1 & 1 \\ 0 & 0 & 3 & 1 & 3 & 1 \\ 0 & 1 & 1 & 6 & 0 & 0 \\ 0 & 2 & 0 & 0 & 4 & 2 \\ 0 & 2 & 1 & 0 & 2 & 3 \end{bmatrix}$

SVC(C=1)
Precision: [0.875 0.36363636 0.4 1. 0.4 0.375]
Accuracy: 0.5416666666666666
Confusion Matrix: $\begin{bmatrix} 7 & 1 & 0 & 0 & 0 & 0 \\ 1 & 4 & 1 & 0 & 1 & 1 \\ 0 & 1 & 2 & 0 & 3 & 2 \\ 0 & 1 & 1 & 6 & 0 & 0 \\ 0 & 2 & 0 & 0 & 4 & 2 \\ 0 & 2 & 1 & 0 & 2 & 3 \end{bmatrix}$

SVC(C=10)
Precision: [0.875 0.5 0.4 1. 0.5 0.44444444]
Accuracy: 0.6041666666666666
Confusion Matrix: $\begin{bmatrix} 7 & 0 & 0 & 0 & 1 & 0 \\ 1 & 4 & 1 & 0 & 1 & 1 \\ 0 & 2 & 2 & 0 & 2 & 2 \\ 0 & 0 & 1 & 6 & 0 & 1 \\ 0 & 1 & 0 & 0 & 6 & 1 \\ 0 & 1 & 1 & 0 & 2 & 4 \end{bmatrix}$

SVC(C=100)
Precision: [0.875 0.5 0.4 1. 0.5 0.44444444]
Accuracy: 0.6041666666666666
Confusion Matrix: $\begin{bmatrix} 7 & 0 & 0 & 0 & 1 & 0 \\ 1 & 4 & 1 & 0 & 1 & 1 \\ 0 & 2 & 2 & 0 & 2 & 2 \\ 0 & 0 & 1 & 6 & 0 & 1 \\ 0 & 1 & 0 & 0 & 6 & 1 \\ 0 & 1 & 1 & 0 & 2 & 4 \end{bmatrix}$

SVC(C=1000)
Precision: [0.875 0.5 0.4 1. 0.5 0.44444444]
Accuracy: 0.6041666666666666
Confusion Matrix: $\begin{bmatrix} 7 & 0 & 0 & 0 & 1 & 0 \\ 1 & 4 & 1 & 0 & 1 & 1 \\ 0 & 2 & 2 & 0 & 2 & 2 \\ 0 & 0 & 1 & 6 & 0 & 1 \\ 0 & 1 & 0 & 0 & 6 & 1 \\ 0 & 1 & 1 & 0 & 2 & 4 \end{bmatrix}$

SVC()
Precision: [0.875 0.36363636 0.4 1. 0.4 0.375]
Accuracy: 0.5416666666666666
Confusion Matrix: $\begin{bmatrix} 7 & 1 & 0 & 0 & 0 & 0 \\ 1 & 4 & 1 & 0 & 1 & 1 \\ 0 & 1 & 2 & 0 & 3 & 2 \\ 0 & 1 & 1 & 6 & 0 & 0 \\ 0 & 2 & 0 & 0 & 4 & 2 \\ 0 & 2 & 1 & 0 & 2 & 3 \end{bmatrix}$

SVC(kernel='linear')
Precision: [0.85714286 0.2 0.4 0.5 0.8 0.5]
Accuracy: 0.5
Confusion Matrix: $\begin{bmatrix} 6 & 2 & 0 & 0 & 0 & 0 \\ 1 & 2 & 3 & 1 & 0 & 1 \\ 0 & 0 & 4 & 3 & 0 & 1 \\ 0 & 1 & 2 & 5 & 0 & 0 \\ 0 & 2 & 0 & 1 & 4 & 1 \\ 0 & 3 & 1 & 0 & 1 & 3 \end{bmatrix}$

SVC(kernel='poly')
Precision: [0.66666667 0.33333333 0.42857143 0.53846154 0.66666667 0.5]
Accuracy: 0.5416666666666666
Confusion Matrix: $\begin{bmatrix} 8 & 0 & 0 & 0 & 0 & 0 \\ 3 & 1 & 2 & 1 & 0 & 1 \\ 1 & 0 & 3 & 3 & 0 & 1 \\ 0 & 0 & 1 & 7 & 0 & 0 \\ 0 & 1 & 0 & 2 & 2 & 3 \\ 0 & 1 & 1 & 0 & 1 & 5 \end{bmatrix}$

SVC(kernel='sigmoid')
Precision: [0.8 0.11111111 0.375 0.54545455 0.5 0.4]
Accuracy: 0.4375
Confusion Matrix: $\begin{bmatrix} 4 & 2 & 0 & 1 & 0 & 1 \\ 1 & 1 & 3 & 2 & 0 & 1 \\ 0 & 0 & 3 & 2 & 2 & 1 \\ 0 & 1 & 1 & 6 & 0 & 0 \\ 0 & 3 & 0 & 0 & 5 & 0 \\ 0 & 2 & 1 & 0 & 3 & 2 \end{bmatrix}$

SVC(gamma='auto')
Precision: [0.625 0.16666667 0.26666667 0.44444444 0.57142857 0.66666667]
Accuracy: 0.4166666666666667

Confusion Matrix: $\begin{bmatrix} 5 & 2 & 0 & 1 & 0 & 0 \\ 2 & 1 & 5 & 0 & 0 & 0 \\ 1 & 0 & 4 & 3 & 0 & 0 \\ 0 & 0 & 4 & 4 & 0 & 0 \\ 0 & 1 & 1 & 1 & 4 & 1 \\ 0 & 2 & 1 & 0 & 3 & 2 \end{bmatrix}$

SVC(gamma=1)
Precision: [0.85714286 0.22222222 0.375 0.85714286 0.44444444 0.375]
Accuracy: 0.5
Confusion Matrix: $\begin{bmatrix} 6 & 2 & 0 & 0 & 0 & 0 \\ 1 & 2 & 3 & 0 & 1 & 1 \\ 0 & 0 & 3 & 1 & 2 & 2 \\ 0 & 1 & 1 & 6 & 0 & 0 \\ 0 & 2 & 0 & 0 & 4 & 2 \\ 0 & 2 & 1 & 0 & 2 & 3 \end{bmatrix}$

SVC(gamma=0.1)
Precision: [0.625 0.25 0.30769231 0.38461538 0.57142857 0.66666667]
Accuracy: 0.4375
Confusion Matrix: $\begin{bmatrix} 5 & 0 & 0 & 3 & 0 & 0 \\ 2 & 1 & 4 & 1 & 0 & 0 \\ 1 & 0 & 4 & 3 & 0 & 0 \\ 0 & 0 & 3 & 5 & 0 & 0 \\ 0 & 1 & 1 & 1 & 4 & 1 \\ 0 & 2 & 1 & 0 & 3 & 2 \end{bmatrix}$

SVC(gamma=0.02)
Precision: [0.625 0.16666667 0.26666667 0.44444444 0.57142857 0.66666667]
Accuracy: 0.4166666666666667
Confusion Matrix: $\begin{bmatrix} 5 & 2 & 0 & 1 & 0 & 0 \\ 2 & 1 & 5 & 0 & 0 & 0 \\ 1 & 0 & 4 & 3 & 0 & 0 \\ 0 & 0 & 4 & 4 & 0 & 0 \\ 0 & 1 & 1 & 1 & 4 & 1 \\ 0 & 2 & 1 & 0 & 3 & 2 \end{bmatrix}$

SVC(gamma=0.001)
Precision: [0.625 0.16666667 0.26666667 0.44444444 0.57142857 0.66666667]
Accuracy: 0.4166666666666667
Confusion Matrix: $\begin{bmatrix} 5 & 2 & 0 & 1 & 0 & 0 \\ 2 & 1 & 5 & 0 & 0 & 0 \\ 1 & 0 & 4 & 3 & 0 & 0 \\ 0 & 0 & 4 & 4 & 0 & 0 \\ 0 & 1 & 1 & 1 & 4 & 1 \\ 0 & 2 & 1 & 0 & 3 & 2 \end{bmatrix}$