LAB 9 - Support Vector Machines

In this lab, we are going to implement support vector machines (SVM). We will perform a classification task (unlike other labs, which were regression tasks) using `sklearn`'s `SVC` class. The details of the class, its instantiators, parameters and functions can be seen in detail here:
https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

We will use the .csv file attached to the assignment in Blackboard. It is a collection of tennis match results with statistics for each player. Now, we are going to try to predict which player won the match (player 1 or player 2, denoted by 0 or 1) based on each player's number of aces.

- (10 pts) Our input (X matrix) is going to consist of the columns "ACE.1" (number of aces of Player 1) and "ACE.2" (number of aces of Player 2). Our Y is going to be the column called "Result".

  We are going to use the first 100 samples as training data, and the rest as test data. Divide your data accordingly.

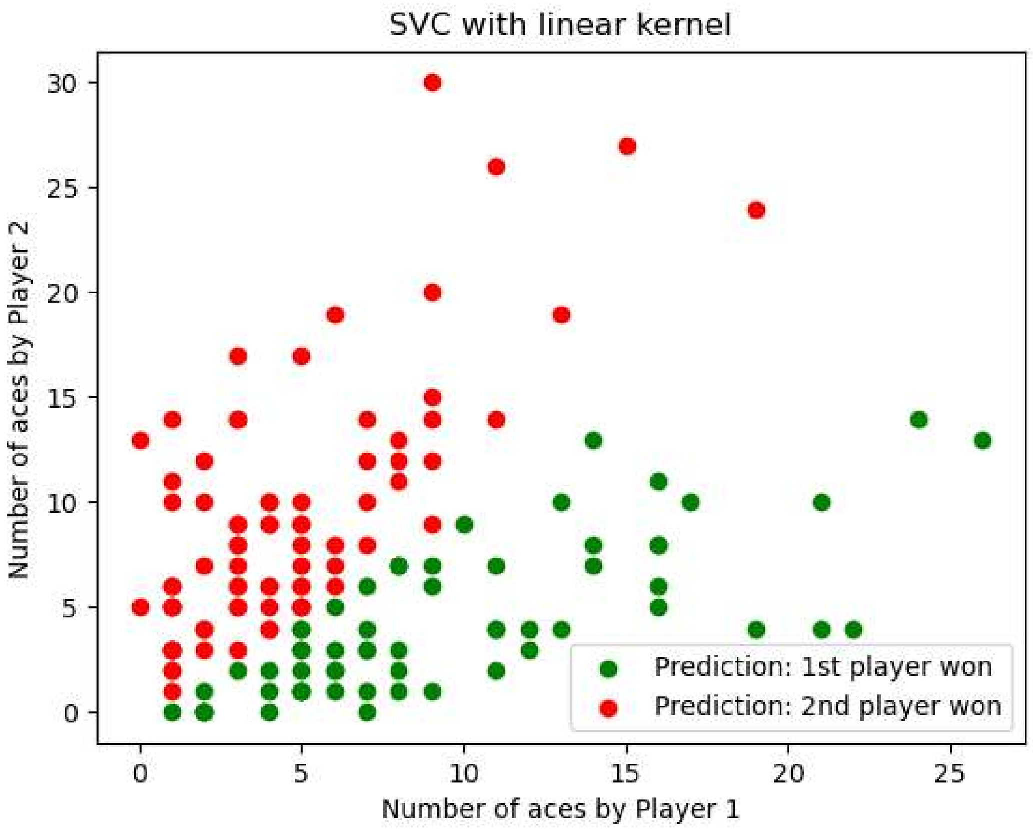
- (60 pts) We are going to implement 3 different `SVC`s and plot their results. Each `SVC` will have a different kernel function:
  - Linear (`kernel = 'linear'` in `SVC` parameters)
  - Polynomial (`kernel = 'poly'` in `SVC` parameters)
  - Radial (`kernel = 'rbf'` in `SVC` parameters) (**Important note**: when creating this `SVC` object, please also add the extra parameter `gamma=auto`).

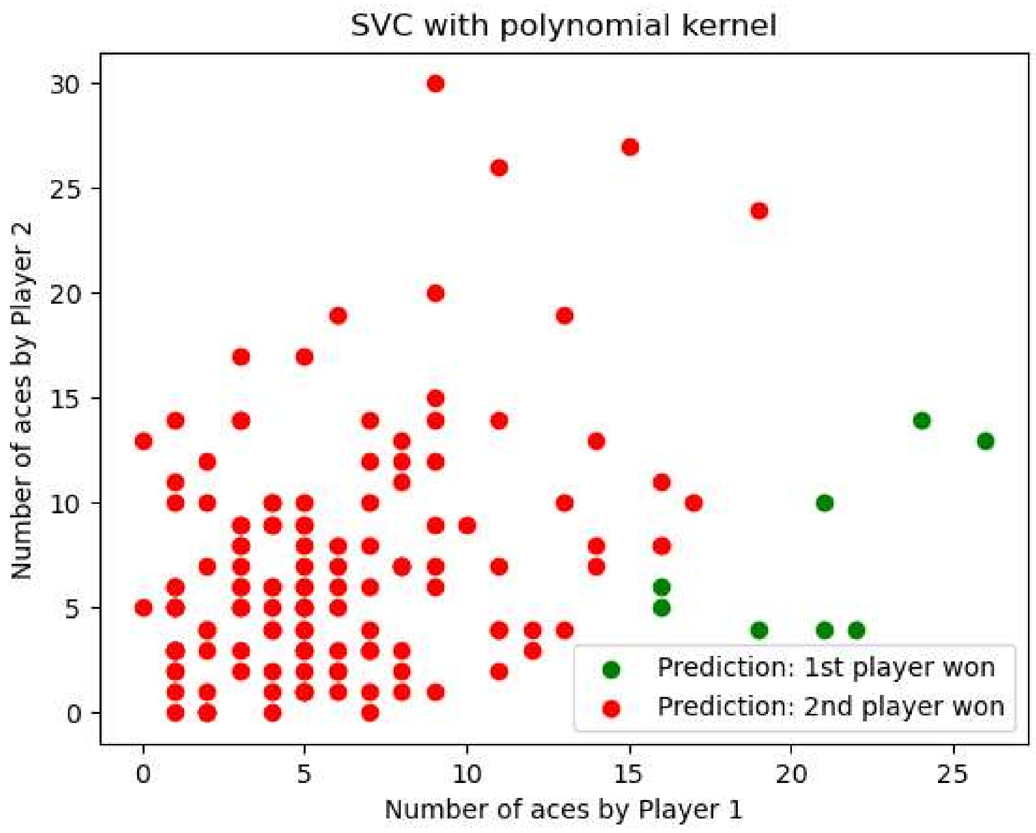
  Follow the instructions below:

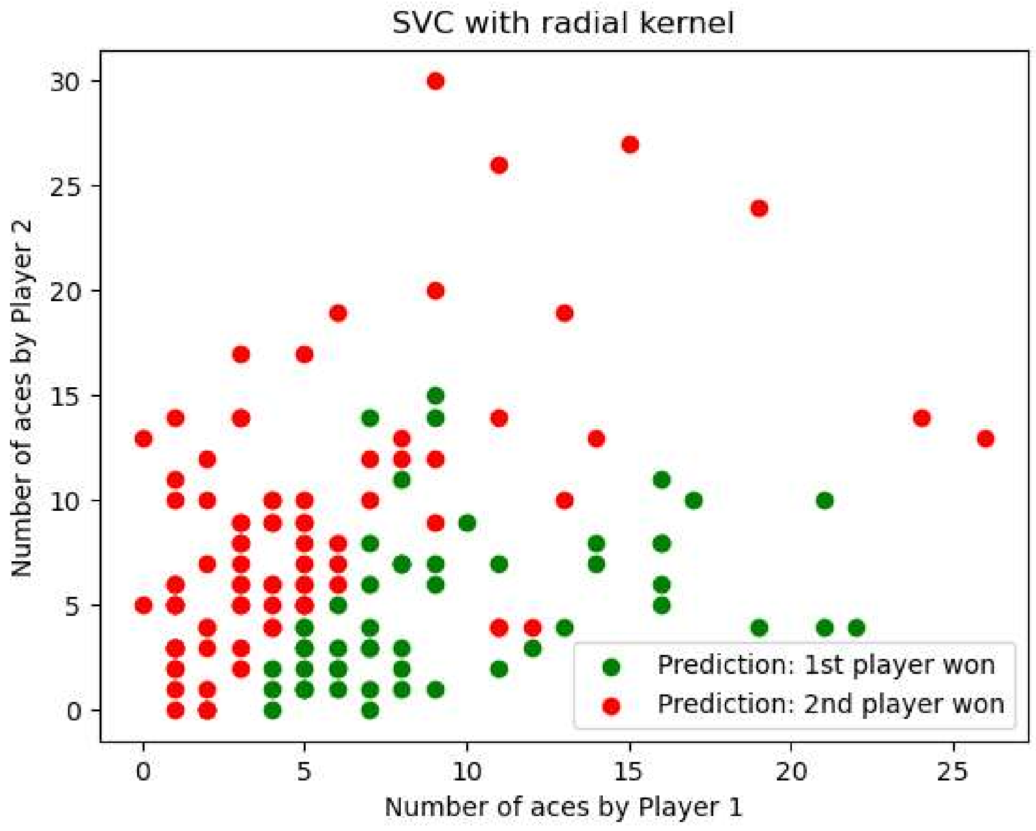
1- Create an `SVC` object.
2- Call the `fit()` function with the training data.
3- Call the `predict()` function with the test input, and store the predictions.
4- The prediction values will consist of 0's and 1's. For each element of the predictions, do the following:
  - If the prediction value is 0, store the index of this element in an array.
  - If the prediction value is 1, store the index of this element in a different array.
5- Extract the rows of your test input using the indices you stored in step 4. You can simply use the arrays (from step 4) directly as indices. Do this for both index arrays in step 4 and store the result for both separately.

6- Repeat steps 1 to 5 for a different `SVC` object. In the end, you should have 6 different matrices with 2 columns.

- (30 pts) The plotting: You will create 3 different plots in 3 different figures (which correspond to the 3 different `SVC` kernel functions). In each figure, you will create 2 different scatter plots. These 2 scatter plots will be drawn using 2 matrices you took from the test input (in step 5 above). Simply call the scatter function, use the 1st column of the matrix as your x-axis, and the 2nd column of the matrix as your y-axis. Draw each scatter plot in a different color. Also, again, implement a legend. Here's the desired output:




SVC with polynomial kernel
Number of aces by Player 2
Number of aces by Player 1
0
5
10
15
20
25
30
Prediction: 1st player won
Prediction: 2nd player won


SVC with radial kernel
Number of aces by Player 2
Number of aces by Player 1
0
5
10
15
20
25
30
Prediction: 1st player won
Prediction: 2nd player won

Continuing with the insight section:

In this lab, we tested 3 different kernel functions for support vector classifiers. You can easily observe the result to see how these functions behave.

There is not much to discuss here (Aside from how support vectors work but if you are unfamiliar with them, consider referring to past courses or the reading material), but we should mention that this was the first <u>classification</u> task we undertook. A classification task is different from a regression task in which the predictions are <u>limited</u> to discrete and finite values (not necessarily numerical values!). Here, our predictions could be only 0 or 1. We used the column “Result” as the output vector, which only included values 0 or 1. The predictions therefore are shaped based on the output vector.

Please notice that we didn’t implement any form of statistics in this lab. You might want to check error scores to compare kernel functions with each other.

Also notice that we haven’t evaluated which predictions were correct, but you can add this functionality with some simple additions to your code.

`SVC`s have many parameters to play with, you can check the documentation referred to above to study what they mean. Like previous labs, please feel free to play with some of those parameters to better understand their effects on the result.