REPORT
ORI LEIBOVITZ
205793896

In this code I test by myself the average of success for each algorithm by taking the training set, and by cut it down to :

```
X of Train = xArray[:int(((total/ 5) * 4))]
x of Test = xArray[int(((total / 5) * 4)):]
y of Train = yArray[:int(((total / 5) * 4))]
y of Test = yArray[int(((total / 5) * 4)):]
```

and after that I could run the code with test to check the algorithms.

<u>Perceptron</u> -
Epochs value: 100
Learning rate : 0.10
Why : I choose: I choose a 100 as iterations because I wanted to be sure that we don't have underfitting but also I didn't get to much iterations so we do not have overfitting for perceptron.

In addition, after testing the algorithm with various values I figured out that the best learning rate for the algorithm is 0.10 → for this rate I got the most stable percentage of success (avg of 60%) after running the code with value ranges between 0 and 1 in 0.1 intervals.

<u>SVM -</u>
Epochs value: 50
Learning rate: 0.10
Lambda: 0.2
Why: I choose as before, I choose a 50 as iterations because I wanted to be sure that we don't have underfitting but also I didn't get to much iterations so we do not have overfitting for SVM.
For the SVM, 100 iterations made it to be little bit overfitting, so I chose to cut it by half and then I got the best avg of success.

In addition, after testing the algorithm with various values I figured out that the best learning rate for the algorithm is 0.10 and the lamabda of 0.2 → for this rate I got the most stable percentage of success (avg of 61%) after running the code with value ranges between 0 and 1 in 0.1 intervals for both.

<u>PA -</u>
Epochs value: 100
Why: I choose a 100 as iterations .
Because epochs is the only parameter in the algorithm that we can control on, I wanted to be sure that we don't have underfitting but also I didn't get to much iterations so we do not have overfitting for SVM.

Normalization

I used the minMax normalization
I ran the code with both normalizations (minMax and z-score), but I choose to use min max because it is more safety to use (after we check that max != min) as we see below:

$$v' = \frac{v - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A$$

and have better result in avg of success.