MACHINE LEARNING & BIG DATA

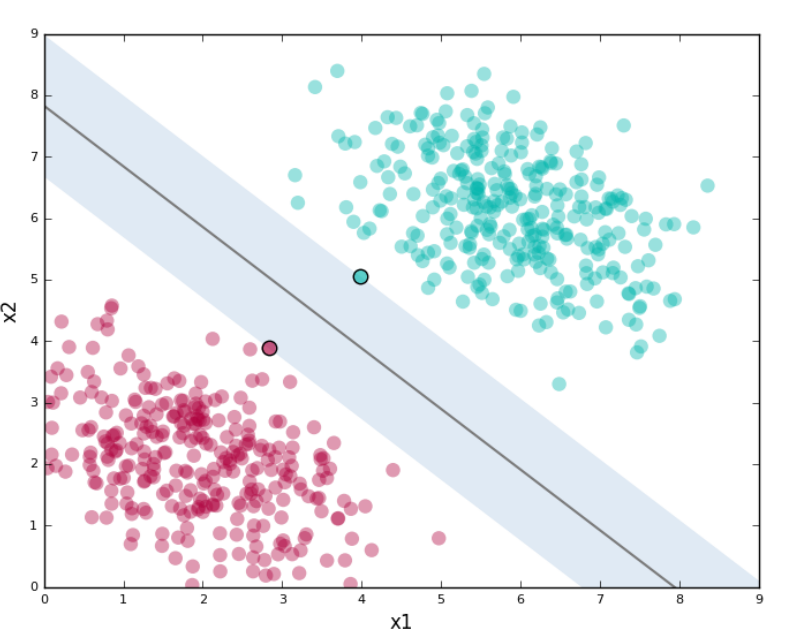
Introduction

SVM vs Logistic Regression

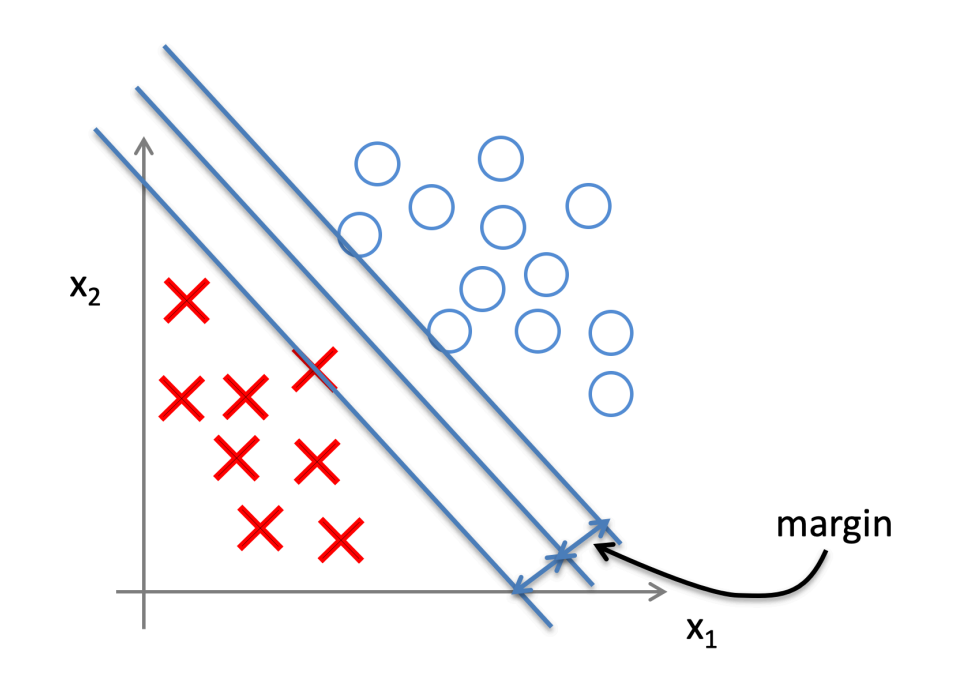
* Support vector machine (SVM) is another way of supervised learning
* It is really similar to the logistic regression
  + They both make classifications by seperating inputs (classification)
* The key difference lays out at “how they classify” them
  + Logistic regression:
    - It has a model (which classifies the inputs) and tries to improve it by trial and error
      * It makes predictions and updates it’s weights according to the result
  + SVM:
    - Unlike logistic regression, it doesn’t have a model
    - It (!!) directly (!!) makes a “boundary”
      * It doesn’t “use a model and then take inputs and makes some predictions and improves itself…”
      * It directly “looks to the inputs” and “how (inputs from different classes) they are distributed” and DIRECTLY PLOTS A BOUNDARY
      * And then improves the drawn boundary…
    - Another difference is that, (unlike to logistic regression) the boundary of SVM is not a “line” but a field
    - Final little difference, SVM is computationally more expensive

SVM Boundaries

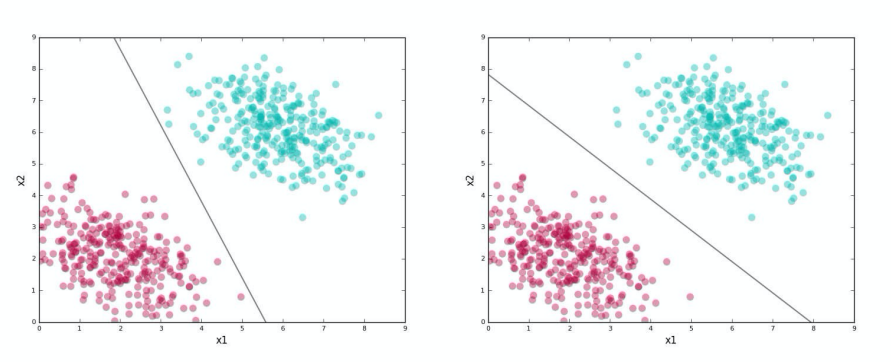
* SVM boundaries are like fields
  + Closest elements of different classes are used as “vectors” to determine the “end-points” of the SVM boundary
    - These vectors are called “Support Vectors”



* + Also, this field is extended with a “margin” (symbolized with C –regularization term- at kernel)
    - It is the “distance between 2 closest support vector”



* + If the margin is too small, it is “very selective” at classification…
    - It has high variance (overfitting)
    - The graph at the left has high variance
      * 2 support vectors are really close to the line which creates boundary
  + If the margin is too large, it is “too simple” at classification…
    - It has high bias (undefitting)
      * Even though we would change inputs, it would still tell us that they belong to the class they belong
    - The graph at the right has high bias
      * 2 support vectors are really far from the line which creates boundary



Kernel

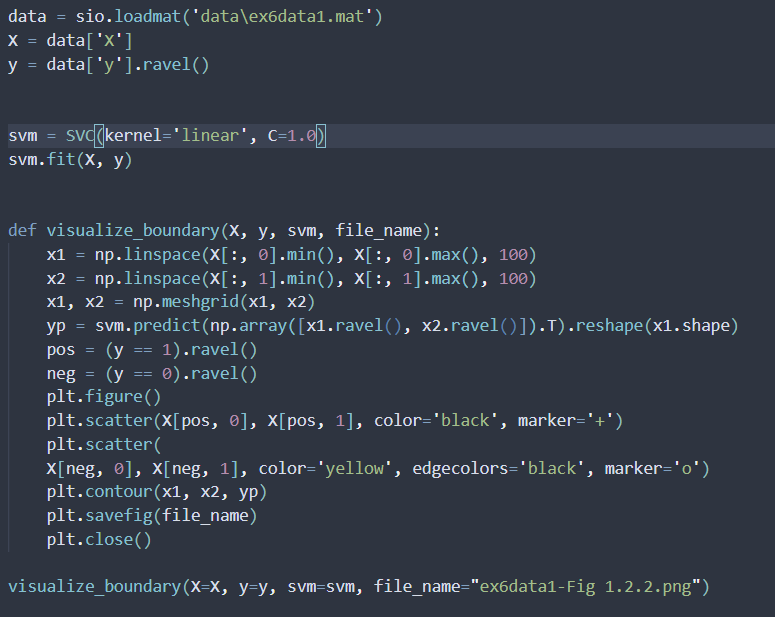
* Kernel is what is used to train SVM (to make boundaries)
  + When working with “non-linear boundaries” (since SVM is computationally really expensive) kernel is used
    - It “increases the dimension size of the feature vector” to make it non-linear
* It has a cost function and iteratively updates a boundary and calculates the cost of that boundary and continues to improve the boundary (until the optimal boundary is reached)
  + It focuses on “maximizing margin” while “minimizing classification errors”
* Here is how kernel is able to plot a decision boundary
  + 1) It doesn’t take all inputs, but the ones which represents “the inputs that belong to that class best” -> landmarks
    - We don’t take an anomaly as the representitive
    - And when we take “a point (which is reaaallyy similiar to other ones)”, then “there is no need to take the other points”
      * Because it is like giving the same input (it is almost the same with other inputs, because it is too similiar)
    - Kernel evaluates this by its “similiarity” function
  + 2) It increases the dimension size of the “feature vector”
  + 3) It finds a boundary (from transformed feature space)
  + 4) It uses cost function to evaluate the boundary
  + 5) Repeats all 4 steps until optimal boundary is reached

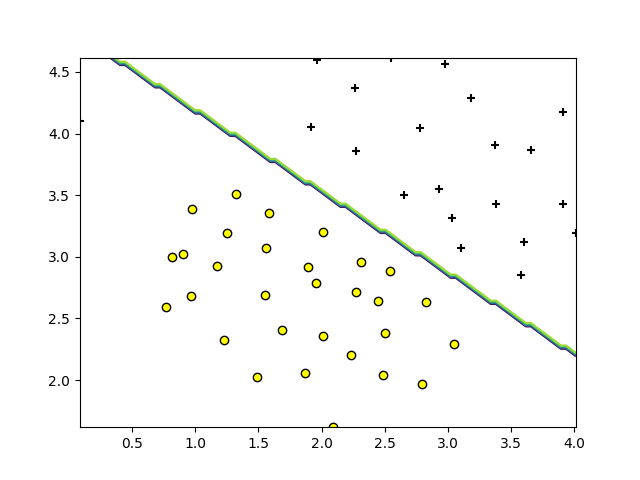
Exercise 1.1

* Since it is only about plotting the data and you will see the plotted data, I won’t put the plotted data’s image

Exercise 1.2 (Both SVM’s are trained with a “linear kernel”)

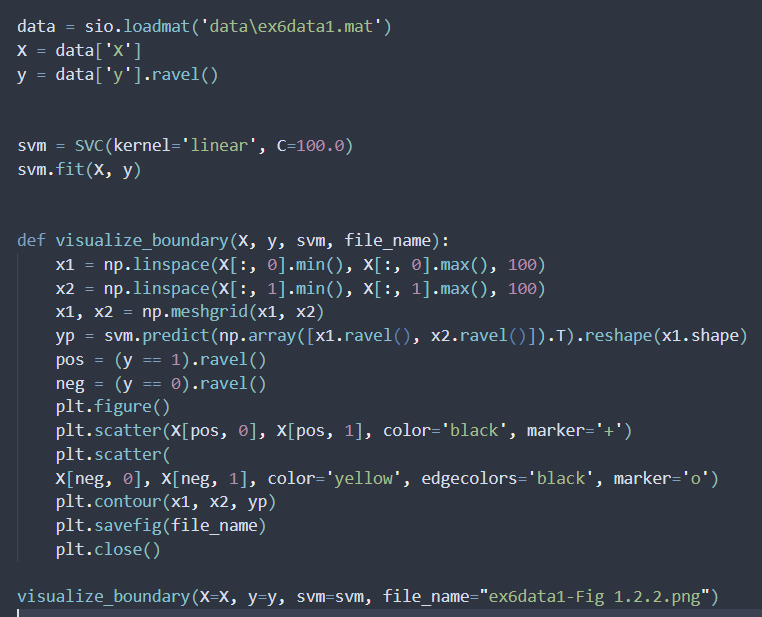
* Graph & function of data1
  + Input matrix X (which is the feature matrix –or vector-, which contains samples) is trained by svm.predict(…) function
    - A person instance can have “value 2.0m” for height feature and “70kg” for weight feature
    - Or it can have “value 1.75m” for height feature and “92 kg” for weight feature
  + Each row is an example, and each column symbolizes a feature of that example

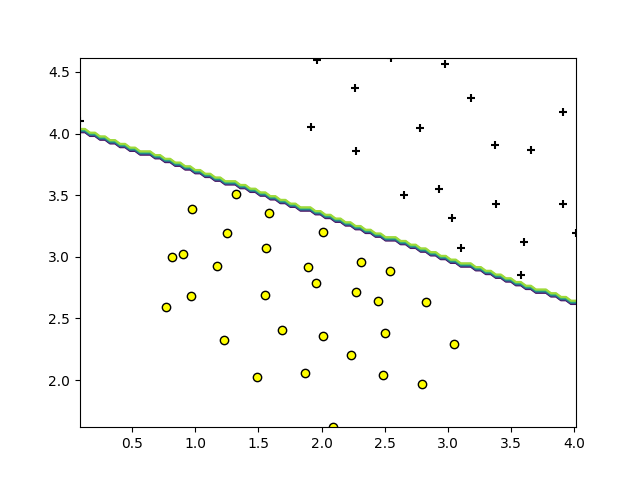




* Graph & function of data 2

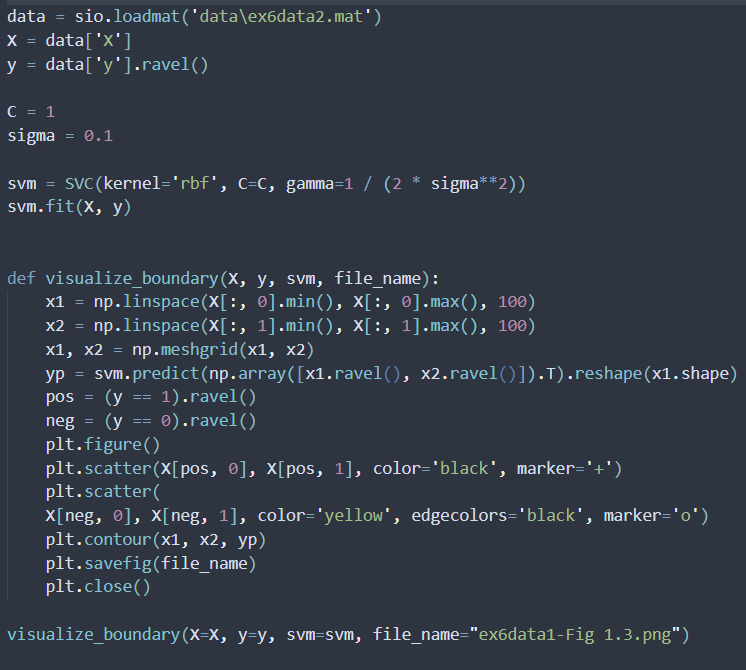
Same function, only C value is changed to 100

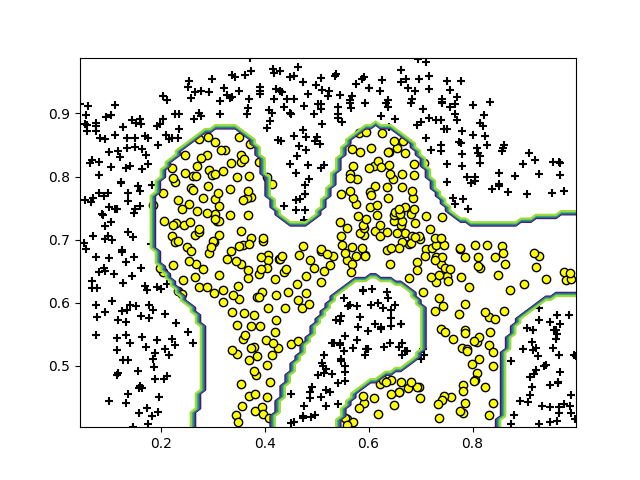




Exercise 1.3

* It uses exaclty the same functions
* Only difference is that “Gaussian Kernel” is used here
  + For the given file, a “non-linear boundary” was needed to correctly classify

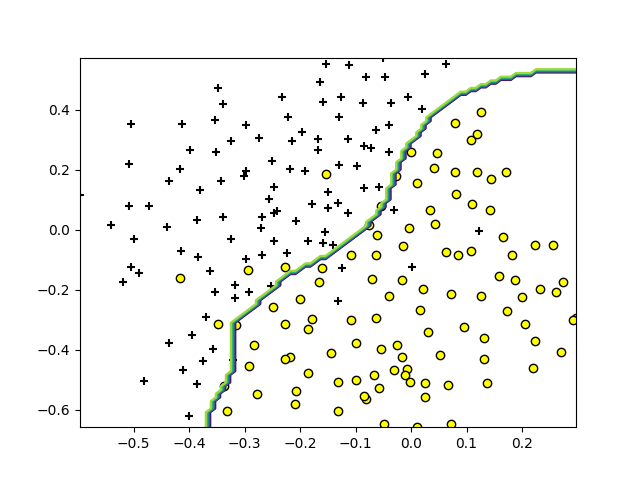




Exercise 1.4

* Although it might look too complex, what we are doing actually is still the same thing from the previous examples
  + We create a svc instance
  + And then train it with kernel
* The only difference here is that, we are “re-training (from scratch)” for each C and sigma values

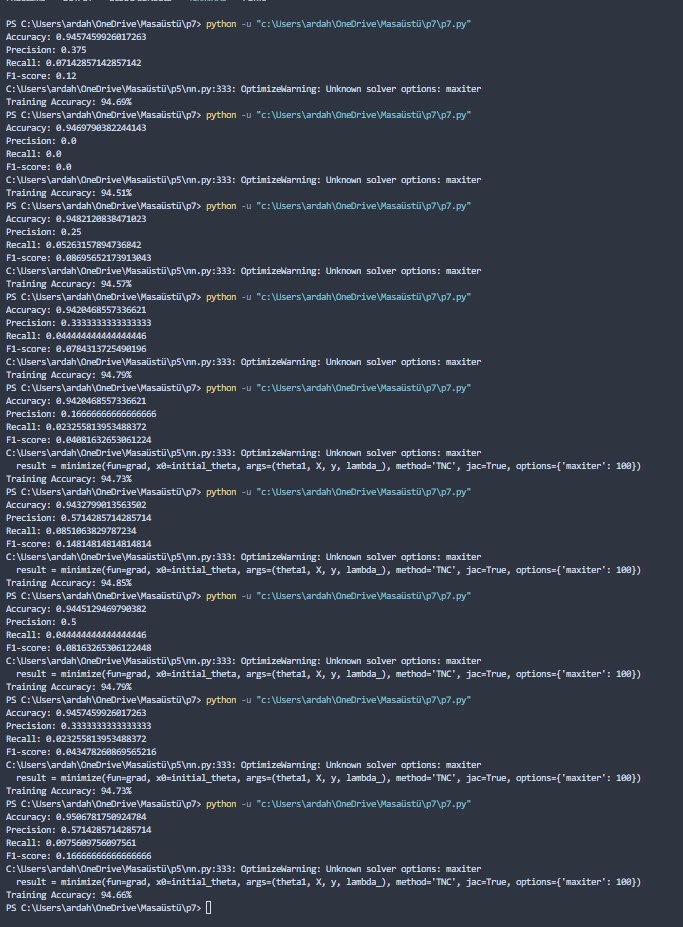




Testing

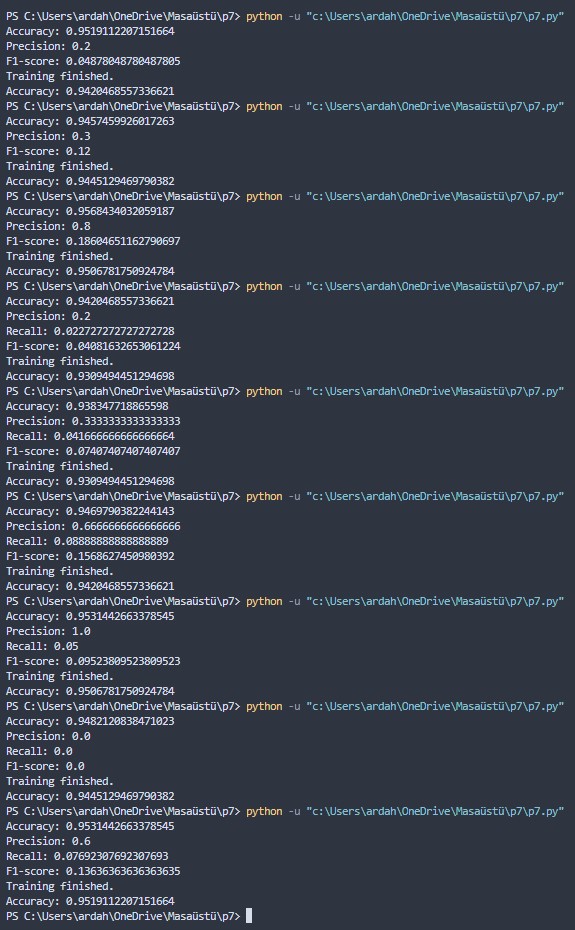
* Contents of testing
  + SVM vs NN (week 5)
  + SVM vs NN (pytorch) (ReLU)
  + SVM vs NN (pytorch) (sigmoid)
  + SVM vs Logistic Regression (without hyperparameter tuning)
  + SVM vs Logistic Regression (with hyperparameter tuning)

SVM vs NN (week 5)



* Upper results belongs to the SVM while lower results belong to the NN
  + SVM
    - Generally gets an accuracy between 0.94-0.95
      * Results are generally a bit lower than SVM but the “highest accuracy rate of SVM” is higher than the “highest accuracy of NN”
  + NN
    - Generally gets an accuracy between 0.945-0.95
* The reason I believe NN Works better is because:
  + 1) It is similar to logistic regression
    - NN can be also described as “using logistic regression multiple times”
      * NN also performs better than logistic regression as well
  + 2) It uses a model and trains that model with input all the time (instead of just creating new boundaries)

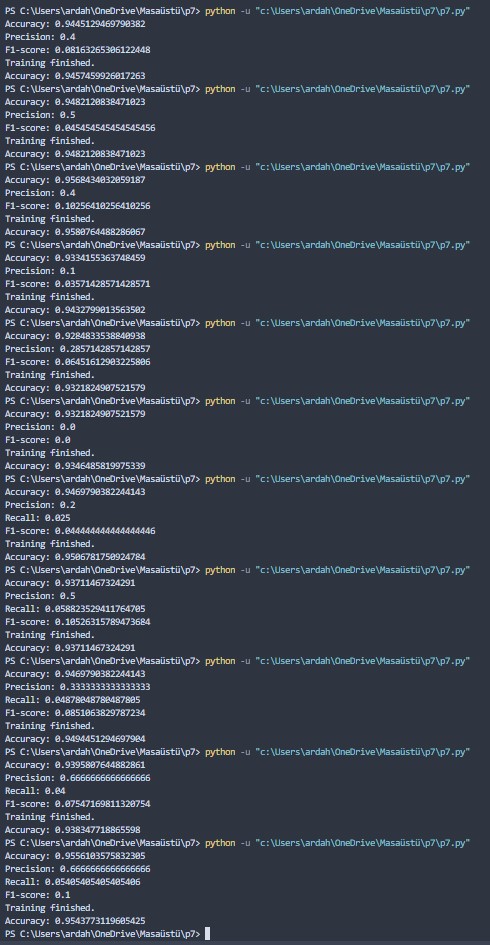
SVM vs NN (pytorch) (ReLU)



* Again, the results are close to each other…
* But, surprisingly this time “SVM performed better than NN (pytorch) (ReLU)
  + I believe, the reason behind this is I used “ReLU” as activation function
  + The “NN (week 5)” had used “sigmoid” as activation function

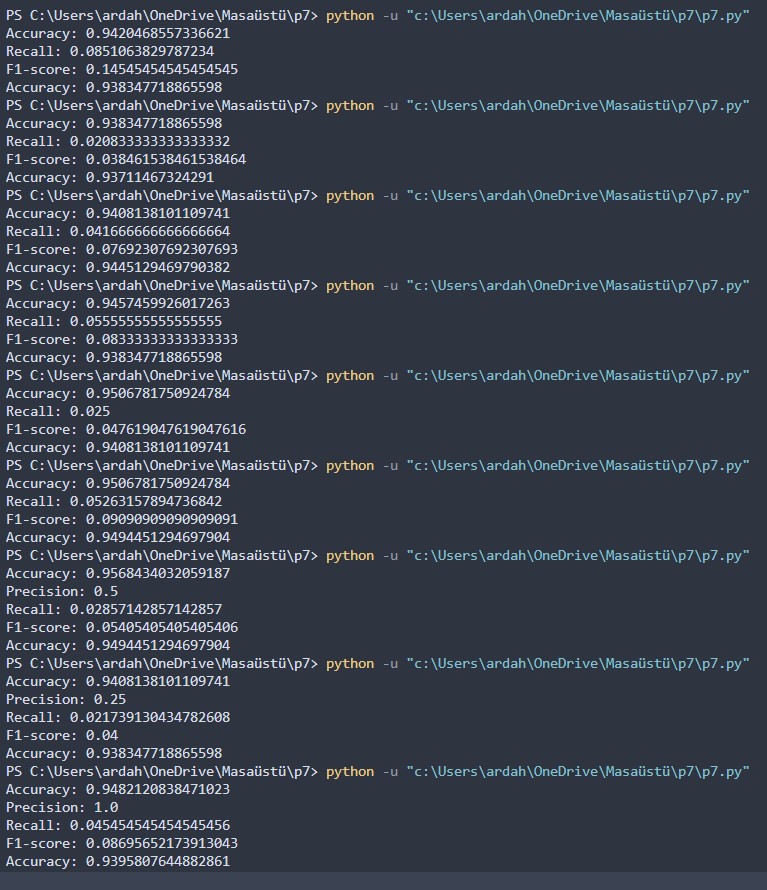
SVM vs NN (pytorch) (sigmoid)

* Like “NN (week 5)” it perfored mostly slightly better than the SVM
  + And this function is almost same with NN (week 5)



SVM vs Logistic Regreession (without hyperparameter tuning)

* SVM mostly performed much better than Logistic Regression…
  + SVM was worse than NN but here it is better than Logistic Regression
    - Because Log. Reg. is more simpler than NN
      * NN uses multiple Log. Reg.s
    - Also it is not regularized and hyper-parameter tuned



SVM vs Logistic Regression (without hyperparameter tuning)

* Interestigly, mostly Logistic Regression is much better than SVM in this case
  + It is because we hyperparameter tuned
    - I wasn’t able to change the polynomial degree because the email data is too big to run
  + And the case log. Reg. performed worse, is I believe that only because “I didn’t made ‘alpha search’ deep enough”
    - Because there are sometimes where both models had reached exactly the same accuracy
      * Because search for alpha was deep enough

