CSE512 Spring 2021 - Machine Learning - Homework 4

Name: Sabrina Margetic

Solar ID: 109898930

Net ID Email Address: sabrina.margetic@stonybrook.edu

Names of people whom you discussed the homework with: None (Only the TA)

1 Question 1: Support Vector Machines

1.1 Linear Case:

$LOOCV = \frac{1}{n}\sum_{i=1}^{n} Error_i$
Let the error associated with a point be 0-1 error.

Removing a Support Vector Data Point:
A support vector is a point that is necessary for correctly creating the SVM hyperplane. Therefore, removing a Support Vector can result in misclassification or an error of 1.

Removing a Non-Support Vector Data Point:
Removing a non-support vector data point has no effect on the SVM. It would still be correctly classified since the SVM would not change. Therefore, the error associated with it is 0.

As an upper bound we can assume a worst case scenario in which, for every support vector left out, they are missclassified and therefore, result in an error of 1. From such, the equation for LOOCV becomes:
$LOOCV = \frac{1}{n}(\sum_{i=1}^{m} 1 + \sum_{i=1}^{n-m} 0)$
where the first summation accounts for support vectors and the later accounts for non support vectors.
$\therefore LOOCV = \frac{1}{n}(m + 0) = \frac{m}{n}$
Since this is the worst case scenario when all Support Vectors are misclassified, we can say that the true value of LOOCV is:
$\boxed{LOOCV \leq \frac{m}{n}}$
The upper bound is proven.

1.2 General Case:

In the general case for this problem, the input space gets converted into feature space. Within this features space, the problem becomes linearly separable. Therefore, within this space that is where classification happens and where the SVM model is built. This is where we will categorize points as either support vectors or non support vectors. Since this general case gets converted to a similar linear case as that above, the same bounds hold for error.

2 XGBoost

2.1

For this part of the assignment, we were instructed to train XGBoost using the entire set of training data. Therefore, the hyper-parameters used were the default values.

When Training Data was Used for Prediction:

Accuracy: 90.43%
The associated confusion matrix for this can be seen in Fig. 1.

When Test Data was Used for Prediction:

Accuracy: 87.00%
The associated confusion matrix for this can be seen in Fig. 2.
There is clearly in a decrease in accuracy for when test data is used. But, this is to be expected.

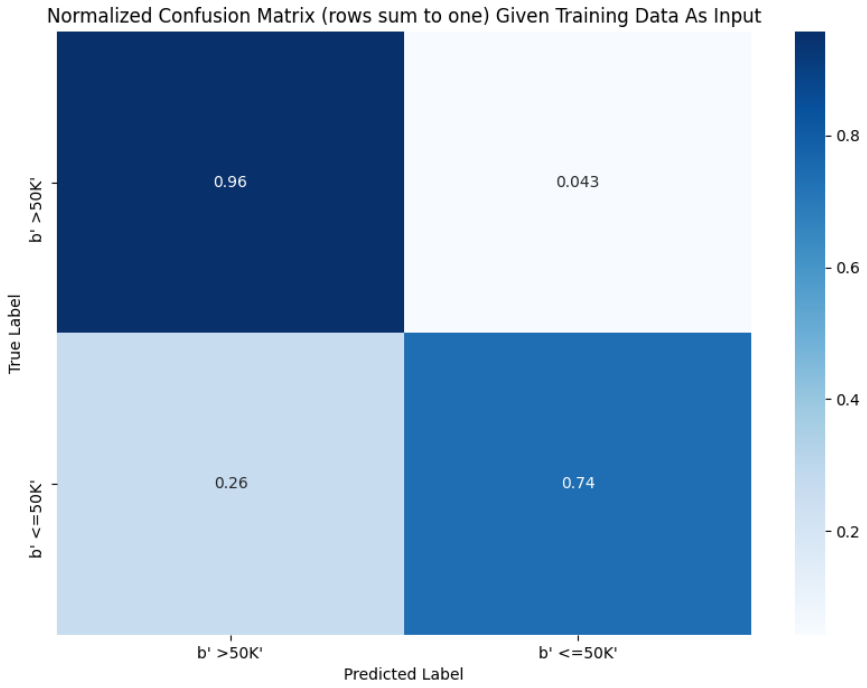

Figure 1

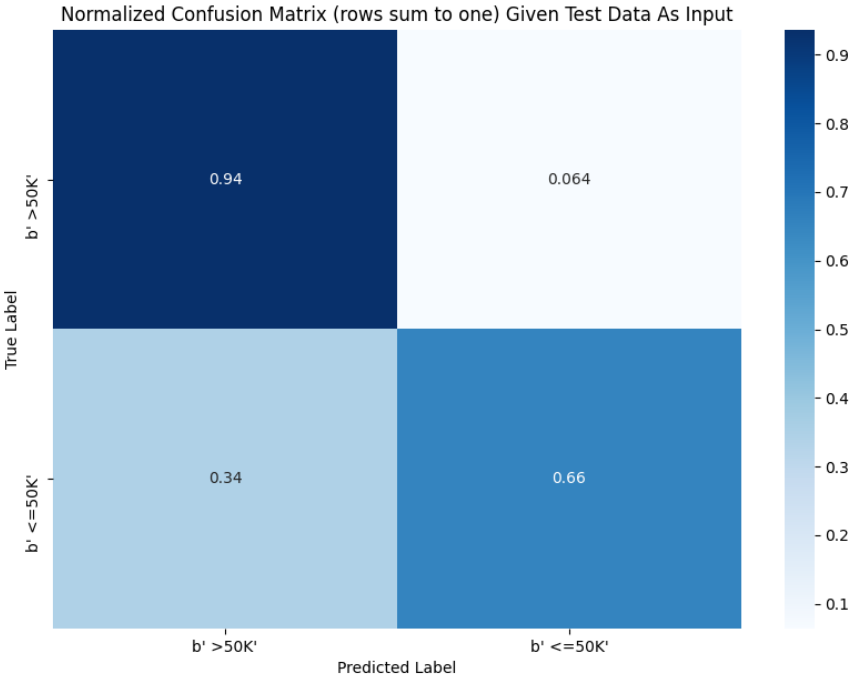

Figure 2

2.2 Hyper-parameters

The following hyper-parameters were trained and there optimal values are as given.

Hyper-parameter	Ideal Values
max_depth	5
max_child_weight	3
subsample	0
colsample_bytree	.96
reg_alpha	.66

The link: https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/ was used as a general tutorial on how to do this. The method used the library, GridSearchCV in order to search for optimal values. The method for quantifying error was changed to MAE so as to not be as sensitive to outliers.

The best cross-validation accuracy was: 87.3226460291137%. The improvement in accuracy in this scenario is negligible.
The confusion matrix produced can be seen in Fig. 3. As you can see, there is almost no difference as the scenario when the parameters were not tuned. However, this can be taken to be a coincidence rather than the norm.

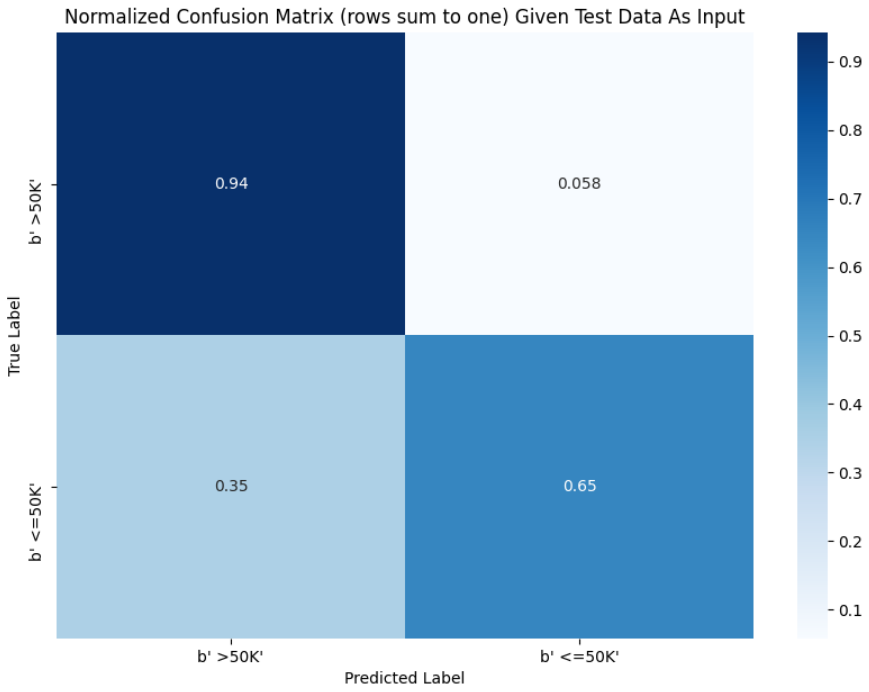

Figure 3

3 SVM for Object Detection

3.1 SVM Classifier

For this part of the assignment, a simple classifier, using python library sklearn.svm.LinearSVC was created. Validation data was used for prediction. The precision recall curve is shown in Fig. 4.
The average precision was calculated to be:

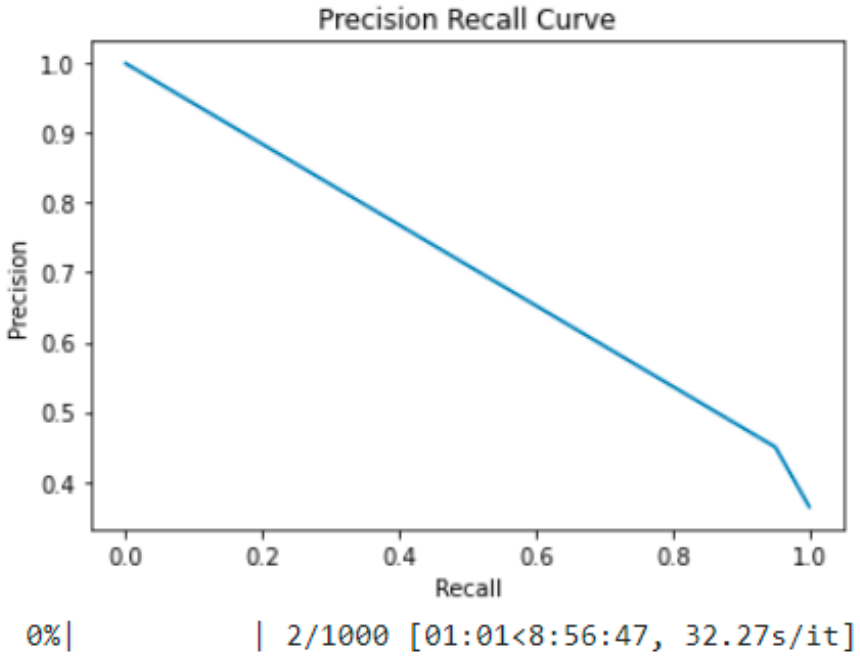

Figure 4

3.2 Hard Negative Mining Algorithm

For this problem the pseudo code provided was followed. Additionally, objective values were stored upon each iteration. The objective function use was $\frac{||w||^2}{2}$ since a linear kernel was utilized.
The Optimization Values Were: [0.12057919699510243, 0.12491387645088808, 0.11947577392919508, 0.11932678665212763, 0.1202370837475727, 0.1181257022938873, 0.12065720959458469, 0.12069755363537946, 0.1227338511759044, 0.12044161679669979, 0.12324901682251448]
The Average Precision Values Were: [0.39126569413798834, 0.45353767258729993, 0.40631731624696565, 0.38967221670536056, 0.3857113745136088, 0.42553370176213845, 0.3795618236114343, 0.3793673480693289, 0.3820039296688995, 0.3692614060721033, 0.40166735873758785]
The corresponding optimization graph and average precision graph can be seen in Figures 5, and 6, respectively. There are some errors with this part of the assignment. But, it unfortunately took me longer than expected, so I wasn't able to work it out.

3.3 Part 3

The code was submitted to the link provided.

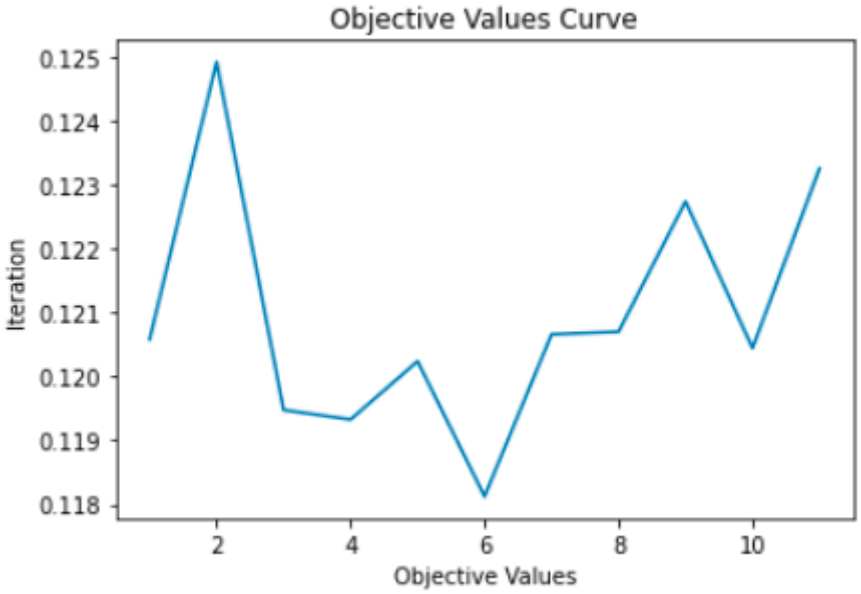

Figure 5

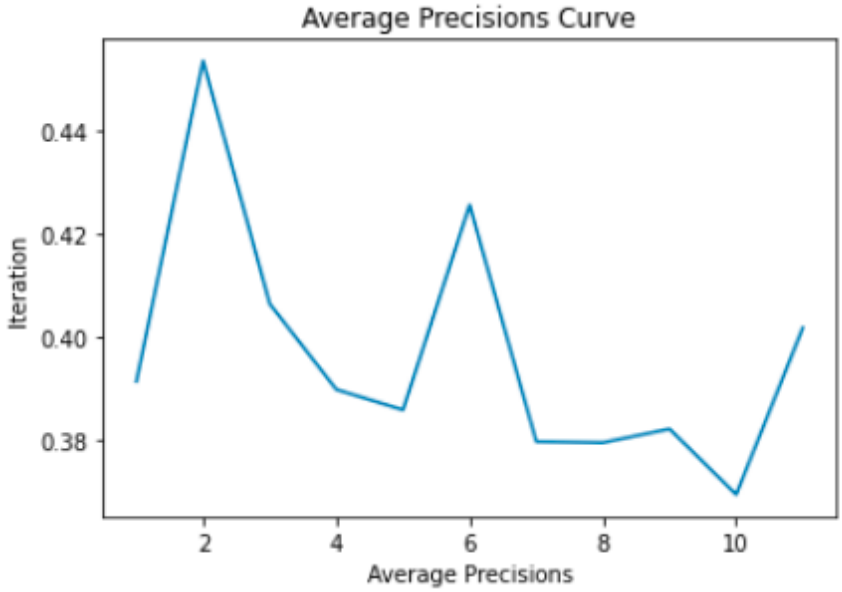

Figure 6