CSE4088 Introduction to Machine Learning

Homework 4

Huzeyfe Ayaz
150119700

In this homework, Support Vector Classifier (SVC) model is imported from Sci-kit Learn (sklearn.svm) module. Numpy is used to create and manipulate array operations. Matplotlib is used to plot the results in question5. Counter class from collections library is imported to count the number of C values in question 7. Lastly, cross_val_score method is implemented from sklearn.model_selection library to handle the cross validation easily.

The code starts with implementing required modules, loading dataset and splitting data into train and test portion.

Since we are going to use SVM classifier with same parameters for questions 2, 3 and 4, SVM is implemented before question 2. Default parameters of SVC are as follows, C=1, degree=3, kernel='rbf', and gamma='scale'. Since polynomial kernel, C=0.01, Q=2 and no-scaling are expected as hyper-parameters in question, parameters are set as specified in question. gamma parameter which is the kernel coefficient is set as 1. one_vs_rest function is implemented to transform labels into binary targets by given class name. For instance, if 2 is given, 1 will be the label for specified class and 0 for others. So that, the problem turns into binary classification instead of multi-class classification.

Question 2 & 3

For question 2 and 3, the choices of the questions have been traversed using for loop and new_targets yielded from one_vs_rest function is used to fit the SVM model. Then, in-sample and out-sample errors are found. Highest E_{in} is obtained in 0 vs all option as ~0.12% and lowest E_{out} is obtained in 1 vs all option as ~0.014% for question 2 and 3 respectively.

Output: `{0: 0.11905088465231106, 2: 0.10026059525442327, 4: 0.08942531888629818, 6: 0.09107118365107666, 8: 0.07433822520916199}`

Output: `{1: 0.014264161294746948, 3: 0.09024825126868742, 5: 0.07625840076807022, 7: 0.08846523110684405, 9: 0.08832807570977919}`

Question 4

In question 4, one_vs_rest function is used 2 times for the class 0 and 1. Then, SVM classifier is fit with training data and transformed train targets. len(model.support_) gives the number of support vectors used in model. After all, absolute differences of number of support vectors are printed out.

Output: `1879`

Question 5

In question 5, train and test data is filtered for the sake of all questions that require only one_vs_five targets. | (bitwise or) allows us to write a query for numpy arrays to filter data. The zeroth column of both train and test data gives us the targets. For example,
`train_data[:, 0] == 1` yields list of boolean values that 1 or True when the target is 1 and 0 or False otherwise. Therefore, the result of query `(train_data[:, 0] == 1) | (train_data[:, 0] == 5)` will be the list of binary numbers 1 when the target is 1 or 5 and 0 otherwise.

Then, the C parameter of SVM is updated inside the for loop and number of support vectors, in sample errors and out of sample errors are stored in lists. Figure 1 shows the result of the values stored in lists.

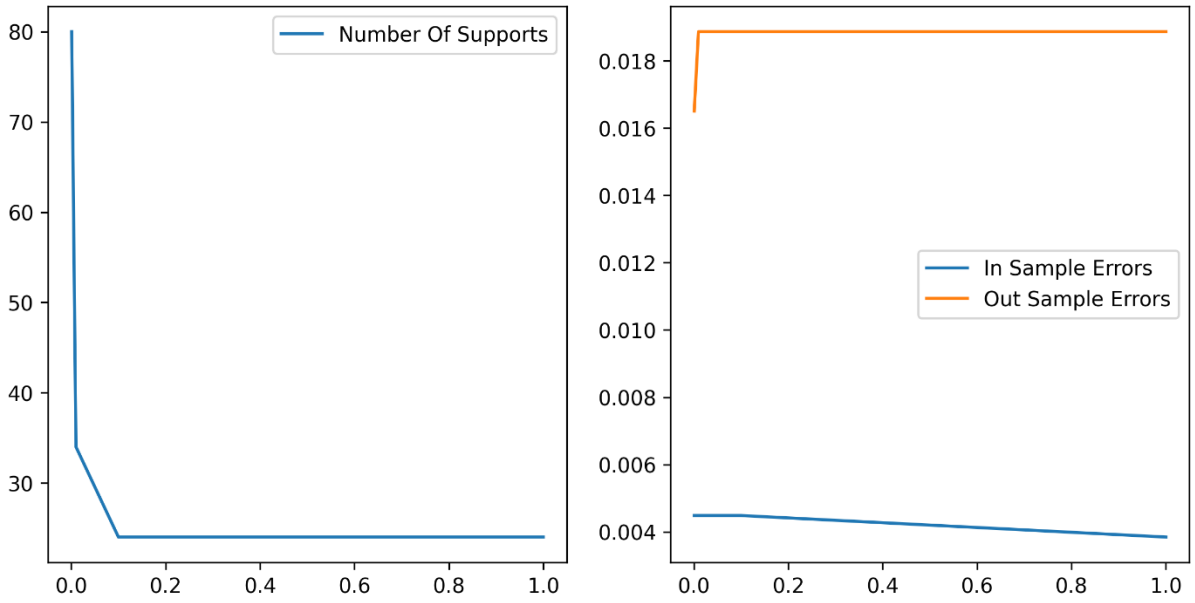

Figure 1: Lineplot of Number of Support Vectors, Insample and Out of Sample Errors

The result of the values are also printed out to console as follows:

Output: `C Values: [0.001, 0.01, 0.1, 1]`

`In Sample Errors: [0.00448…, 0.00448…, 0.00448…, 0.00384…]`

`Out Sample Errors [0.01650…, 0.01886…, 0.01886…, 0.01886…]`

`Num. Of Sup. Vec.: [80, 34, 24, 24]`

Question 6

Likewise, C values are updated in each iteration and results are stored in the lists. However, in this case degree parameter is chosen as 5 since Q=5. Figure 2 shows the result of the values stored in lists.

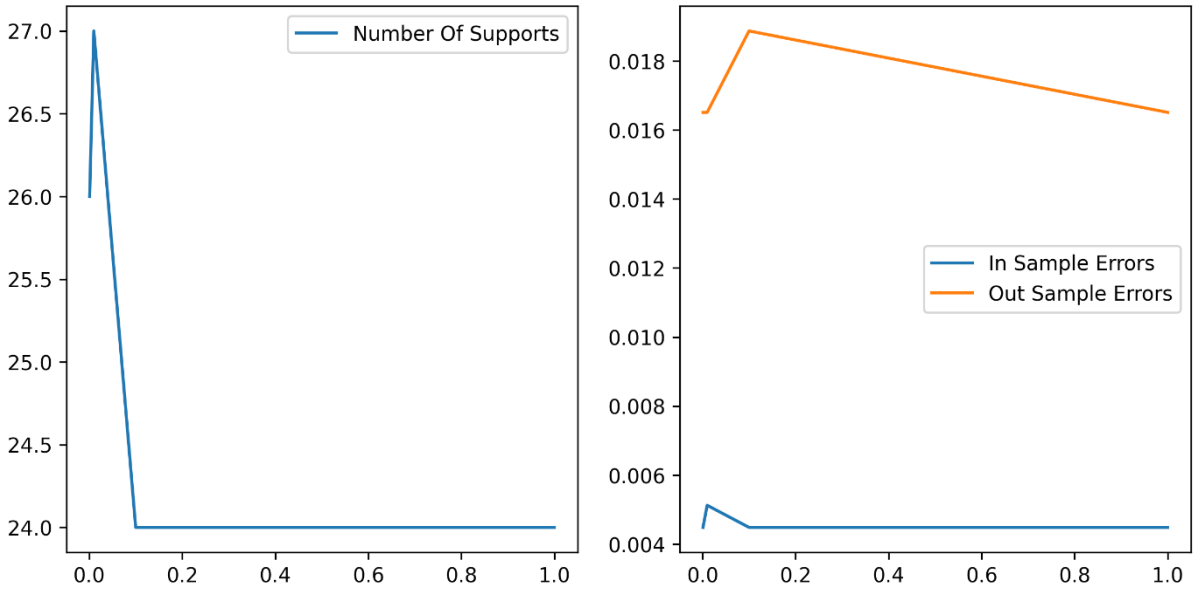

Figure 2: Lineplot of Number of Support Vectors, Insample and Out of Sample Errors

The result of the values are also printed out to console as follows:

Output:
```
C Values: [0.001, 0.01, 0.1, 1]
In Sample Errors: [0.00448…, 0.00512…, 0.00448…, 0.00448…]
Out Sample Errors [0.01650…, 0.01650…, 0.01886…, 0.01650…]
Num. Of Sup. Vec.: [26, 27, 24, 24]
```

Question 7

Nested for loop is implemented for question 7. The first loop iterates over the C values while second loop is repeating the operation 100 times for each C value. cross_val_score method in sklearn automatically handles the cross-validation operation, then, fits and evaluates the model with a splitted portions. Since we want to obtain errors, the result of the scores are subtracted from 1. After that, scores are converted into numpy array format in the shape of (5, 1000) which each row corresponds to a C value and each column is a fold. Numpy's argmin function helps us to find the minimum index among either rows or columns with specified axis (in this case 0 since we want to get min C value). Counter class from collections module automatically counts the number of elements given list and stores in dictionary. As seen in the output, 1st index which is 0.001 is selected more often.

Output:
```
Counter({0: 177, 1: 405, 2: 242, 3: 112, 4: 64})
```

Question 8

For question 8, the mean of each row is taken.

Output: `array([0.01081288, 0.00499669, 0.00457333, 0.00472142, 0.00497089])`

Question 9

Similar path is followed in question 9, but in this case C values were different. As mentioned before, argmin gives the minimum index. Hence C_values[min_index] will give the min C value.

Output: `E in: 1000000.0`

Question 10

Output: `E out: 100`