Amirkabir University of Technology
(Tehran Polytechnic)

Computer Engineering Department

Machine Learning Course
CE5501

Homework 4

Student
Reza Sajedi
400131072
r.sajedi@aut.ac.ir

Teacher
Dr. Nazerfard

Fall 2022

Table of Contents

Problem 1: Why and how

Part a

For noisy datasets, we should reduce the effects of the noisy data points on the training model by decreasing the variance or adding some bias. For the former purpose, the Bagging methods are preferred.

Part b

It's based on a theoretical foundation that indicates sampling with replacement and then building an ensemble reduces the variance of the model without increasing the bias. The same theoretical property is not true if we sample without replacement, because sampling without a replacement would result in high variance. Another reason is if for example, we want to build 100 models with a training set that has 200 instances, sampling without replacement dedicates only 2 samples to each model which is impractical.

Part c

If we really have a robust classifier, there is no need to improve it by applying the ensemble methods. A major point in ensemble learning is combining simple and fast, but not-so-good, classifiers and improving the error rate. If we use robust classifiers, the room for improvement becomes smaller yet the computational cost becomes larger, making ensemble methods less interesting. Moreover, a single robust classifier may be easier to interpret.

Part d

The goal of the Bagging method is to decrease the variance of the model and they are good for deep trees. According to the number of features, the trees with max depth 4 are very high bias. The accuracy of each model must be at least a bit larger than 50% to apply ensemble methods. So, the overall accuracy will be so small and it's not acceptable.

Part e

When we have a low dimensional dataset with many samples, applying Bagging methods is better. In Bagging methods, the number of samples from the original dataset should be large enough to capture most of the complexity of the underlying distribution so that sampling from the dataset is a good approximation of sampling from the real distribution (representativity). Also, the size of the dataset should be large enough compared to the size of the bootstrap samples so that samples are not too much correlated (independence). Also, the capability of the Bagging methods to perform in parallel makes that a good choice for large-scale datasets.

When the dataset is high dimensional and has fewer samples, it's better to apply Boosting methods, because by assigning a weight for each sample and adapting the next classifiers to focus on the wrong predicted instances which have higher weights, we can fit a model appropriately while we

have just a few samples. Also, learning the weights for the samples doesn't have a large computational cost in such cases.

Part f

$$\epsilon^t = \sum_{\substack{i \\ \text{Incorrectly classified examples}}} w_i^t$$

$$\alpha^t = \frac{1}{2}\ln\frac{1-\epsilon^t}{\epsilon^t}$$

$$\begin{cases} w_i^{t+1} = \frac{w_i^t}{2} \times \frac{1}{1-\epsilon^t} & h^t(\boldsymbol{x_i}) = y_i \\ w_i^{t+1} = \frac{w_i^t}{2} \times \frac{1}{\epsilon^t} & h^t(\boldsymbol{x_i}) \neq y_i \end{cases}$$

t	$w_{(0.5,1.5)}$	$w_{(1.5,1.5)}$	$w_{(1.5,0.5)}$	$w_{(2.5,1.5)}$	$w_{(2.5,2.5)}$	ϵ	α
1	0.2	0.2	0.2	0.2	0.2	0.2	0.6931
2	0.125	0.125	0.5	0.125	0.125	0.25	0.5493
3	0.0833	0.0833	0.3333	0.25	0.25	0.0833	1.1991

$$h(x) = sgn\left(0.6931 \times h^1(x) + 0.5493 \times h^2(x) + 1.1991 \times h^3(x)\right)$$

Part g

1)

$$f_x = \alpha g_x \rightarrow -2x = \alpha \rightarrow x = -\frac{1}{2}\alpha$$

$$f_y = \alpha g_y \rightarrow -4y = \alpha \rightarrow y = -\frac{1}{4}\alpha$$

$$-\frac{1}{2}\alpha - \frac{1}{4}\alpha - 1 = 0 \rightarrow \alpha = -\frac{4}{3} \rightarrow x = \frac{2}{3}, y = \frac{1}{3}$$

$$f\left(\frac{2}{3}, \frac{1}{3}\right) = \frac{4}{3}$$

2)

$$f_x = \alpha_1 g_{1x} + \alpha_2 g_{2x} \rightarrow 2x = \alpha_1 \rightarrow x = \frac{1}{2}\alpha_1$$

$$f_y = \alpha_1 g_{1y} + \alpha_2 g_{2y} \rightarrow 2y = \alpha_2 \rightarrow y = \frac{1}{2}\alpha_2$$

$$\frac{1}{2}\alpha_1 + 1 = 0 \rightarrow \alpha_1 = -2 \rightarrow x = -1$$
$$\frac{1}{2}\alpha_2 + 1 = 0 \rightarrow \alpha_2 = -2 \rightarrow y = -1$$
$$f(-1,-1) = 2$$

3)

$$f_x = \alpha_1 g_{1x} + \alpha_2 g_{2x} \rightarrow 3x^2 = 2\alpha_1 x \rightarrow x = \frac{2}{3}\alpha_1$$
$$f_y = \alpha_1 g_{1y} + \alpha_2 g_{2y} \rightarrow 3y^2 = 2\alpha_2 y \rightarrow y = \frac{2}{3}\alpha_2$$
$$x^2 - 1 = 0 \rightarrow \frac{4}{9}\alpha_1^2 - 1 = 0 \rightarrow \alpha_1^2 = \frac{9}{4} \rightarrow \alpha_1 = \pm\frac{3}{2} \rightarrow x = \pm 1$$
$$y^2 - 1 = 0 \rightarrow \frac{4}{9}\alpha_2^2 - 1 = 0 \rightarrow \alpha_2^2 = \frac{9}{4} \rightarrow \alpha_2 = \pm\frac{3}{2} \rightarrow y = \pm 1$$
$$f(1,1) = 2$$
$$f(1,-1) = 0$$
$$f(-1,1) = 0$$
$$f(-1,-1) = -2$$

Part h

$$L(\alpha) = -\frac{1}{2}\left(\alpha_1 \begin{bmatrix}1\\1\end{bmatrix} - \alpha_2 \begin{bmatrix}2\\2\end{bmatrix}\right).\left(\alpha_1 \begin{bmatrix}1\\1\end{bmatrix} - \alpha_2 \begin{bmatrix}2\\2\end{bmatrix}\right) + \alpha_1 + \alpha_2 = -\alpha_1^2 - 4\alpha_2^2 + 4\alpha_1\alpha_2 + \alpha_1 + \alpha_2$$

$$\begin{cases} \frac{\partial L(\alpha)}{\partial \alpha_1} = -2\alpha_1 + 4\alpha_2 + 1 = 0 \\ \frac{\partial L(\alpha)}{\partial \alpha_2} = -8\alpha_2 + 4\alpha_1 + 1 = 0 \\ \alpha_1 - \alpha_2 = 0 \end{cases}$$

$$\begin{cases} \alpha_1 - 2\alpha_2 + 1 = 0 \\ \alpha_1 = \alpha_2 \end{cases}$$

$$\alpha_1 = \alpha_2 = 1$$

$$w = \sum_i \alpha_i y_i x_i \rightarrow w = \begin{bmatrix}1\\1\end{bmatrix} - \begin{bmatrix}2\\2\end{bmatrix} = \begin{bmatrix}-1\\-1\end{bmatrix}$$

$$y_i(w.x_i + b) - 1 = 0 \rightarrow \begin{bmatrix}-1 & -1\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix} + b - 1 = 0 \rightarrow b = 3$$

$$w.x + b = 0 \rightarrow w_1x_1 + w_2x_2 + b = 0 \rightarrow x_2 = -\frac{w_1}{w_2}x_1 - \frac{b}{w_2} \rightarrow x_2 = -x_1 + 3$$

Part i

1) $K(x,z) = K_1(x,z) + K_2(x,z)$

K is a symmetric kernel matrix because it's sum of two valid kernels. So, K is positive semi-definite and a valid kernel if for each z:

$$z^T K z = z^T (K_1 + K_2) z = z^T K_1 z + z^T K_2 z \geq 0$$

Thus, we can conclude that K is a valid kernel.

2) $K(x,z) = K_1(x,z) - K_2(x,z)$

K is not necessarily a valid kernel, because it may happen that $z^T K_2 z > z^T K_1 z$ and in such situation the kernel becomes negative.

3) $K(x,z) = \alpha K_1(x,z)$

K is a symmetric matrix, because K1 is a valid kernel and just multiplied by a constant.

$$\begin{aligned} & z^T K z \\ &= z^T (aK_1) z \\ &= a z^T K_1 z \\ &\geq 0 \qquad\qquad a \in \mathbb{R}^+, \quad z^T K_1 z \geq 0 \ \forall z \end{aligned}$$

So, K is positive semi-definite and a valid kernel.

4) $K(x,z) = K_1(x,z)K_2(x,z)$

K is symmetric and a valid kernel:

$$K_1 \text{ is a kernel, so } \exists\ \phi^{(1)} \text{ such that } K_1(x,z) = \left(\phi^{(1)}(x)\right)^T \left(\phi^{(1)}(z)\right)$$

$$K_2 \text{ is a kernel, so } \exists\ \phi^{(2)} \text{ such that } K_2(x,z) = \left(\phi^{(2)}(x)\right)^T \left(\phi^{(2)}(z)\right)$$

$$
\begin{aligned}
K(x,z) &= K_1(x,z)K_2(x,z) \\
&= \sum_i \phi_i^{(1)}(x)\phi_i^{(1)}(z) \sum_j \phi_j^{(2)}(x)\phi_j^{(2)}(z) \\
&= \sum_i \sum_j \phi_i^{(1)}(x)\phi_i^{(1)}(z)\phi_j^{(2)}(x)\phi_j^{(2)}(z) \\
&= \sum_i \sum_j \left[\phi_i^{(1)}(x)\phi_j^{(2)}(x)\right] \left[\phi_i^{(1)}(z)\phi_j^{(2)}(z)\right] \quad \text{def } \psi(\cdot) = \phi^{(1)}(\cdot)\phi^{(2)}(\cdot) \\
&= \sum_{(i,j)} \psi_{i,j}(x)\psi_{i,j}(z) \\
&= \psi(x)^T \psi(z)
\end{aligned}
$$

Part j

Because the newly added data-point may be one of the possible support vectors. So, the calculation of the decision boundary and its margin should be repeated.

We can use the online variants of the SVM. Also, we can use the Stochastic Gradient Descent with the hinge loss and L2 regularization that gives us an SVM model that can be updated online/incremental. We can combine this with feature transforms that approximate a kernel to get similar to an online kernel SVM.

Part k

KKT conditions form the backbone of linear and nonlinear programming as they are necessary and sufficient for optimality in linear programming and convex optimization, and necessary for optimality in non-convex optimization problem. The four necessary KKT conditions is as follows:

Stationarity

For minimizing $f(x)$: $\partial f(x^*) + \sum_{j=1}^{\ell} \lambda_j \partial h_j(x^*) + \sum_{i=1}^{m} \mu_i \partial g_i(x^*) \ni \mathbf{0}$

For maximizing $f(x)$: $-\partial f(x^*) + \sum_{j=1}^{\ell} \lambda_j \partial h_j(x^*) + \sum_{i=1}^{m} \mu_i \partial g_i(x^*) \ni \mathbf{0}$

Primal feasibility

$h_j(x^*) = 0, \text{ for } j = 1, \ldots, \ell$

$g_i(x^*) \leq 0, \text{ for } i = 1, \ldots, m$

Dual feasibility

$\mu_i \geq 0, \text{ for } i = 1, \ldots, m$

Complementary slackness

$\sum_{i=1}^{m} \mu_i g_i(x^*) = 0.$

Problem 2: SVM with custom kernel

The results show that the custom kernel generated with the K2 (Sigmoid) kernel has better performance than the original kernel. The original K1 (RBF) kernel is better than the customized version.

rbf: $\exp(-\gamma\|x - x'\|^2)$, where γ is specified by parameter `gamma`, must be greater than 0.
sigmoid $\tanh(\gamma\langle x, x'\rangle + r)$, where r is specified by `coef0`.

```
Results for K1 (RBF) Kernel:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        14
           1       1.00      0.88      0.93         8
           2       0.89      1.00      0.94         8

    accuracy                           0.97        30
   macro avg       0.96      0.96      0.96        30
weighted avg       0.97      0.97      0.97        30



Results for K2 (Sigmoid) Kernel:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        14
           1       0.00      0.00      0.00         8
           2       0.27      1.00      0.42         8

    accuracy                           0.27        30
   macro avg       0.09      0.33      0.14        30
weighted avg       0.07      0.27      0.11        30



Results for 'K1(gamma=0.0019) + K1(gamma=0.004)' Kernel:
              precision    recall  f1-score   support

           0       1.00      0.93      0.96        14
           1       0.88      0.88      0.88         8
           2       0.89      1.00      0.94         8

    accuracy                           0.93        30
   macro avg       0.92      0.93      0.93        30
weighted avg       0.94      0.93      0.93        30



Results for 'K2(gamma=0.005, coef0=0) * K2(gamma=0.01, coef0=0.1)' Kernel:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        14
           1       1.00      0.62      0.77         8
           2       0.73      1.00      0.84         8

    accuracy                           0.90        30
```

```
   macro avg       0.91      0.88      0.87        30
weighted avg       0.93      0.90      0.90        30
```

Problem 3: SVM outlier detection

Part a

The instruction has been followed.

Part b

One-class SVM is an unsupervised model for anomaly or outlier detection. Unlike the regular supervised SVM, the one-class SVM can be used when the training instances don't have labels. Instead, it learns the boundary for the normal data points and identifies the data outside the border to be outliers.

When training the one-class SVM, there are a few critical hyperparameters.

- *nu* is used to specify the percentage of outliers in the dataset.

- *Kernel* specifies the kernel type. RBF kernel is a commonly used kernel type.

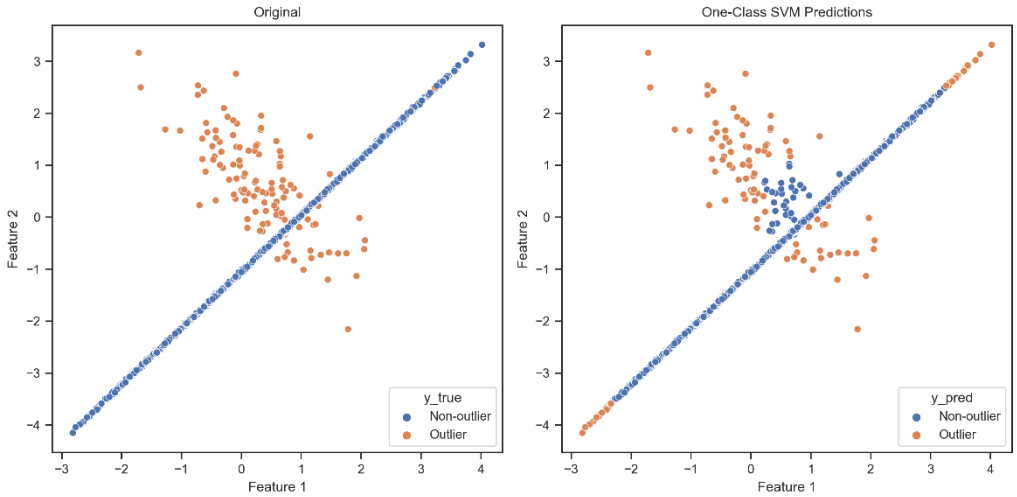
- *gamma* is a kernel coefficient, and it is for 'rbf', 'poly', and 'sigmoid' kernels. When setting it to 'auto', the kernel coefficient is 1 over the number of features.

Part c

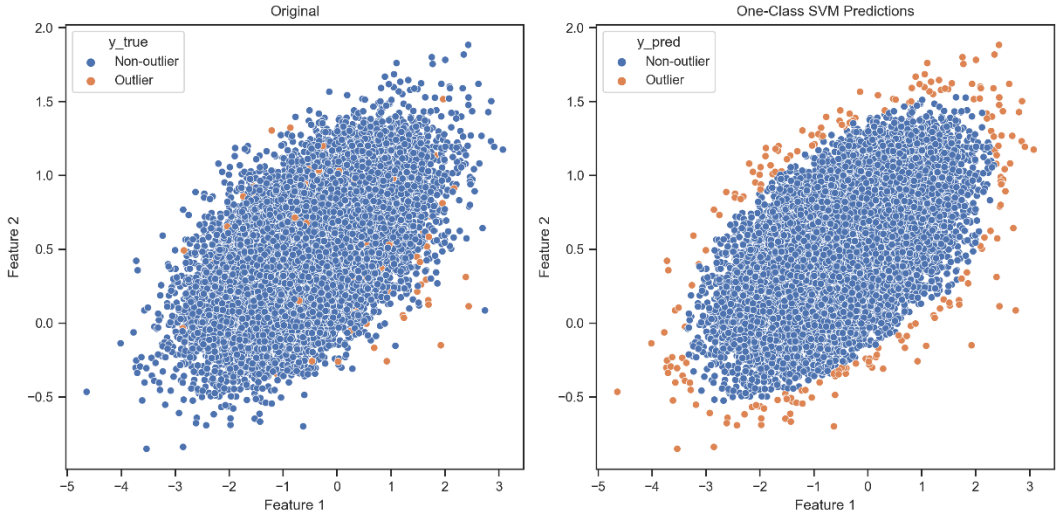
In the anomaly detection task, the class of interest is the outlier. A model which can detect most of the outliers has a good performance. So, the True Positive Rate (TPR) a.k.a. recall is a suitable metric for evaluation.

```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99     19787
           1       0.39      0.40      0.39       213

    accuracy                           0.99     20000
   macro avg       0.69      0.70      0.69     20000
weighted avg       0.99      0.99      0.99     20000
```

Part d



If we set the *random_state* parameter to 0, we get the figure as follows:



Problem 4: Create an anti-malware

Part a

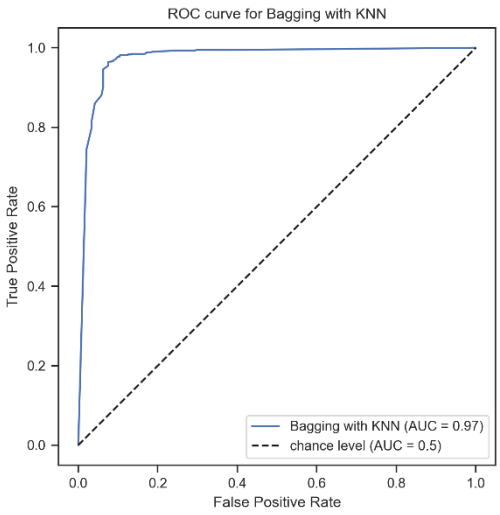
False Negative error is much more important than the other errors. We must try to reduce it as much as possible. Because if we incorrectly consider an instance clean, it may cause irreparable damage to the system. When we incorrectly consider a clean instance as virus, the user can check it and report it as clean and doesn't have much cost like the former.

Part b

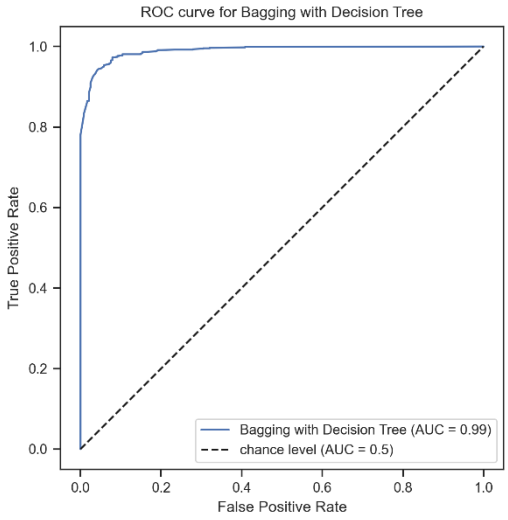
Stratified splitting is used when the dataset is imbalanced. It keeps the distribution of the target variable in each division the same as the original dataset. Without stratifying the performance of the model may be decreased.

Part c

```
Best Parameters for Bagging with KNN:
base_estimator__metric: cosine
base_estimator__n_neighbors: 7
bootstrap_features: true
n_estimators: 5

Results for Bagging with KNN:
Accuracy: 0.9609
Recall: 0.9819
Confusion Matrix:
[[0.1883 0.0249]
 [0.0142 0.7726]]
```

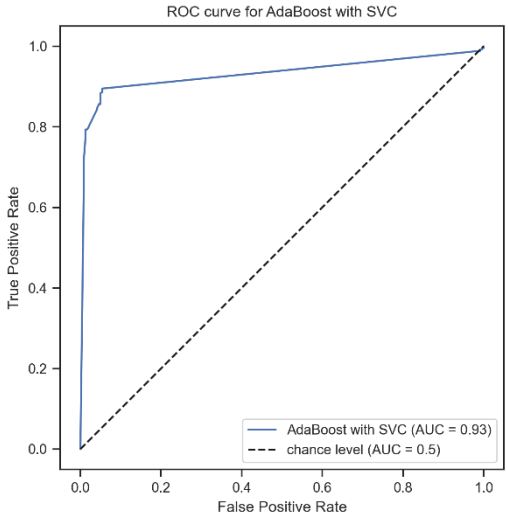



```
Best Parameters for Bagging with Decision Tree:
base_estimator__max_depth: 30
bootstrap_features: true
n_estimators: 55

Results for Bagging with Decision Tree:
Accuracy: 0.96
Recall: 0.974
Confusion Matrix:
[[0.1936 0.0195]
 [0.0204 0.7664]]
```



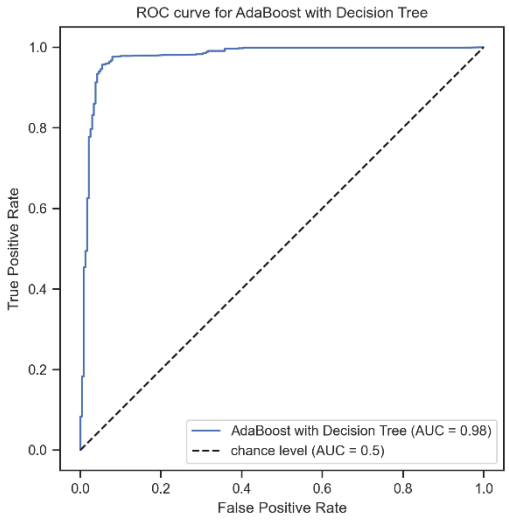
Part d

```
Best Parameters for AdaBoost with SVC:
base_estimator__C: 0.1
base_estimator__kernel: linear
n_estimators: 20

Results for AdaBoost with SVC:
Accuracy: 0.9469
Recall: 0.9438
Confusion Matrix:
[[0.2035 0.0088]
 [0.0442 0.7434]]
```




```
Best Parameters for AdaBoost with Decision Tree:
base_estimator__max_depth: 5
learning_rate: 1.0
n_estimators: 55

Results for AdaBoost with Decision Tree:
```

```
Accuracy: 0.9636
Recall: 0.9752
Confusion Matrix:
[[0.1963 0.0169]
 [0.0195 0.7673]]
```



Part e

The instruction has been followed.

Part f

The instruction has been followed.

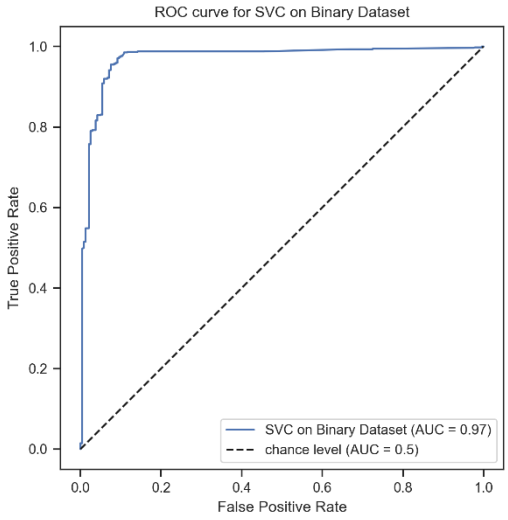
Part g

The instruction has been followed.

Part h

I prefer using SVC on the binary dataset, because it has both higher recall and accuracy scores than the other models. The results of the other models are very similar.

```
Results for SVC on Binary Dataset:
Accuracy: 0.9636
Recall: 0.9853
Confusion Matrix:
[[0.1883 0.0249]
 [0.0115 0.7753]]
```




```
Results for LogisticRegression on Binary Dataset:
Accuracy: 0.9583
Recall: 0.9774
Confusion Matrix:
[[0.1892 0.024 ]
 [0.0178 0.7691]]
```