Machine Learning
Lab 4

In this lab session, we will practice how to implement several evaluation methods for supervised learning. Check the following examples and use these codes to evaluate the model you developed in previous weeks.

1. Accuracy (Classifier)

The accuracy of a classifier is calculated as the ratio of the total number of correctly predicted samples by the total number of samples.

Accuracy metric can be used to evaluate the classifier when the data set is a balanced data set. Accuracy metric should not be used when the data set is imbalanced.

Let us consider a data set with two target classes containing 100 samples out of which 95 samples belong to class 1 and 5 samples belong to class 2. When we try to build a classifier for the above data set, the classifier will be biased to class 1 and will result is predicting all the samples as class 1 samples. This will result in an accuracy of 95%, which is false.

To avoid this mistake accuracy metric should be only used balanced data set.

Now let us look into the code to get the accuracy of a classifier:

```
# Importing all necessary libraries
from sklearn.metrics import accuracy_score
# Calculating the accuracy of classifier
print(f"Accuracy of the classifier is: {accuracy_score(y_test, predictions)}")
```

2. Confusion Matrix (Classifier)

A confusion matrix is an N dimensional square matrix, where N represents total number of target classes or categories. Confusion matrix can be used to evaluate a classifier whenever the data set is imbalanced. Let us consider a binary classification problem i.e. the number of target classes are 2. A typical confusion matrix with two target classes (say “Yes” and “No”) looks like:

	Predicted: NO	Predicted: YES
Actual: NO	True Negative (TN)	False Positive (FP)
Actual: YES	False Negative (FN)	True Positive (TP)

There are four important terms in a confusion matrix:

- True Positives (TP): These are the cases where the predicted “Yes” actually belonged to class “Yes”.
- True Negatives (TN): These are the cases where the predicted “No” actually belonged to class “No”.
- False Positives (FP): These are the cases where the predicted “Yes” actually belonged to class “No”.
- False Negatives (FN): These are the cases where the predicted “No” actually belonged to class “Yes”.

Now let us look into the code to generate and plot the confusion matrix for our breast cancer classifier.

```
# Confusion Martrix
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay

# confusion_matrix funnction a matrix containing the summary of predictions
print(confusion_matrix(y_test, y_pred))

# ConfusionMatrixDisplay function is used to visualize the confusion matrix
ConfusionMatrixDisplay.from_estimator(classifier, X_test, y_test)
plt.show()
```

3. Precision (or Positive Predictive Value) (Classifier)

Precision is the ratio of true positives (TP) by the sum of true positives (TP) and false positives (FP).

$$Precision = \frac{True\ Positives\ (TP)}{True\ Positives\ (TP) + False\ Positives\ (FP)}$$

Let us consider a data set with two target classes (say positive and negative) then precision tells us, out of total predicted positive values how many were actually positive.

Precision should be used based on the use case.

Take an example use case of spam detection. If our model detects a mail as spam which was not actually a spam mail then the user might miss an important mail i.e. here false positives should be reduced.

So, in this use case we need to use precision as a metric to measure the quality of our classifier.

Now let us look into the code to calculate the precision score for our breast cancer classifier:

```
# Importing all necessary libraries
from sklearn.metrics import precision_score


# Calculating the precision score of classifier
print(f"Precision Score of the classifier is: {precision_score(y_test, predictions)}")
```

4. Recall (or Sensitivity or True Positive Rate) (Classifier)

Recall is the ratio of true positives (TP) by the sum of true positives (TP) and false negatives (FN).

$$Recall = \frac{True\ Positives\ (TP)}{True\ Positives\ (TP) + False\ Negatives\ (FN)}$$

```
# Importing all necessary libraries
from sklearn.metrics import recall_score

# Calculating the recall score of classifier
print(f"Recall Score of the classifier is: {recall_score(y_test, predictions)}")
```

4. F1 Score (Classifier)

F1 score should be used when both precision and recall are important for the use case. F1 score is the harmonic mean of precision and recall. It lies between [0,1].

$$F1\ Score = \frac{2 * Precision * Recall}{Precision + Recall}$$

F1 score is derived from F Beta Score. F Beta score is the weighted harmonic mean of precision and recall.

$$F_\beta\ Score = \frac{(1+\beta^2) * Precision * Recall}{(\beta^2 * Precision) + Recall}$$

- If both False Positives (FP) and False Negatives (FN) are important then β = 1.
- If False Positive (FP) is important then β lies between o and 1.
- If False Negative (FN) is important then β > 1.

Now let us look into the code to calculate the f1 score for our breast cancer classifier:

```
# Importing all necessary libraries
from sklearn.metrics import f1_score

# Calculating the F1 score of classifier
print(f"F1 Score of the classifier is: {f1_score(y_test, predictions)}")
```

5. AUC-ROC Curve (Classifier)

AUC-ROC Curve is a performance metric that is used to measure the performance for the classification model at different threshold values. ROC is Receiver Operating Characteristic Curve and AUC is Area Under Curve. The higher the value of AUC (Area under the curve), the better is our classifier in predicting the classes. AUC-ROC is mostly used in binary classification problems.

The ROC curve is plotted between True Positive Rate (TPR) and False Positive Rate (FPR) i.e. TPR on the y-axis and FPR on the x-axis. AUC is the area under the ROC curve. An excellent classifier has an AUC value near 1, whereas a poor-performing classifier has an AOC value near 0. A classifier with an AOC score of 0.5 doesn't have any class separation capacity.

$$TPR = \frac{TP}{TP + FN}$$

True Positive rate

$$FPR = \frac{FP}{FP + TN}$$

False Positive Rate

Now let us look into the code to calculate and plot ROC AUC.

```
# Importing all necessary libraries
from sklearn.metrics import roc_curve, auc

class_probabilities = classifier.predict_proba(X_test)
preds = class_probabilities[:, 1]

fpr, tpr, threshold = roc_curve(y_test, preds)
roc_auc = auc(fpr, tpr)

# Printing AUC
print(f"AUC for our classifier is: {roc_auc}")

# Plotting the ROC
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

6. Activity - Feature Importance

- Apply the above evaluation methods to the model you developed last week.
- Find a method and implement it to show the importance of features from the classification results.