Report: Evaluating Classification Model Performance Using SVC

This is a step-by-step process of evaluating a classification model using key performance metrics and visualization tools in Python. Each step includes an explanation of the function used, its purpose, and the outcomes of the execution, supported by visual outputs and their interpretations.

Unit11_model_Performance_Measurement(

Importing Required Libraries

The code starts by importing libraries from Scikit-learn and Matplotlib to handle data preparation, model training, prediction, and evaluation:
- `confusion_matrix`, `classification_report`, `roc_curve`, `precision_recall_curve`: Used to generate performance metrics.
- `train_test_split`: Splits the dataset into training and testing subsets.
- `SVC`: Support Vector Classifier used as the predictive model.
- `ConfusionMatrixDisplay`, `plt`: To plot the confusion matrix and various curves.

Data Generation and Splitting

Using `make_classification`, we generate a synthetic binary dataset. `train_test_split` is then used to divide it into training and testing sets for model validation.

Model Training with SVC

The Support Vector Classifier (`SVC`) initialized with a radial basis function (RBF) kernel and enable probability estimation. This configuration allows to derive probabilistic outputs necessary for ROC and precision-recall curves.

This setup ensures the model is flexible, accurate, and evaluatable using advanced diagnostics.

It is especially useful in imbalanced datasets where traditional metrics (e.g., accuracy) may be misleading.

Support Vector Classifier (SVC)

- The SVC is a classification algorithm from the scikit-learn library that implements Support Vector Machines (SVM).
- SVM is a supervised learning model used for binary and multiclass classification.
- It works by finding the optimal hyperplane that separates data points of different classes with the maximum margin.

2. Radial Basis Function (RBF) Kernel

- The kernel is a function that maps data into a higher-dimensional space so that it becomes easier to classify with a linear decision boundary.
- The RBF kernel (also known as Gaussian kernel) is the most commonly used and is defined as:

$K(x,x')=\exp(-\gamma\|x-x'\|2)$

Where:

- $K(x,x')$ is the kernel function (the output similarity between two input vectors xxx and x′x'x′).
- $\|x-x'\|2$ is the squared Euclidean distance between two data points xxx and x′x'x′.
- γ\gamma is a hyperparameter that defines how much influence a single training example has. It controls the "spread" of the kernel or how sharply similarity falls off with distance. A small $\gamma \rightarrow$ broader area of influence (many points are considered similar). A Large $\gamma \rightarrow$ narrow area of influence (only very close points are similar). It affects model flexibility:
- Large γ = more complex, may overfit.
- Small γ = simpler, may underfit.

The RBF function

- Works well for non-linear relationships, where the data cannot be separated by a straight line in the original space.
- By using RBF, the classifier becomes capable of learning complex decision boundaries.

Enabling Probability Estimation

- By default, SVC does not return probabilities, only class labels.
- Setting probability=True enables the model to:
    - Use Platt scaling to convert SVM outputs into calibrated probabilities.
    - Return predict_proba() values for each class (between 0 and 1).

The Platt scaling is a post-processing method used to convert the raw outputs (decision scores) of a Support Vector Machine (SVM) into calibrated probability estimates.

SVMs typically output distance scores from the decision boundary (also called margins or decision function values), not probabilities. These scores tell you how confidently a sample is classified but do not represent calibrated probabilities (i.e., actual chances between 0 and 1). Platt scaling fits a logistic regression model to the SVM's decision scores on a held-out validation set.

Why Probabilities Are Necessary

The probability output is required in applications like ROC curves, risk prediction, or medical diagnosis, we need real probabilities. That's where Platt scaling comes in—to transform decision scores into true probability estimates.

- ROC Curve (Receiver Operating Characteristic):
    - Plots True Positive Rate (TPR) vs. False Positive Rate (FPR).
    - Requires confidence scores or probabilities to generate the curve.

Precision-Recall Curve:

- Plots precision vs. recall at different threshold levels.
- Also depends on probabilistic outputs rather than discrete predictions.

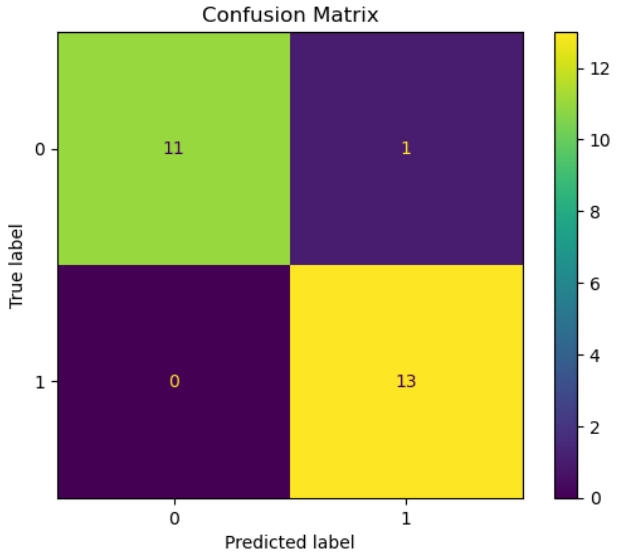
Summary of What It Does

- Initializes a powerful non-linear classifier using the RBF kernel.

- Configures it to output probabilities for each class using probability=True.
- Enables the generation of rich performance metrics like ROC AUC and Precision-Recall, which offer deeper insights into model behaviour beyond simple accuracy.

Predictions and Evaluation

Predictions are made on the test set using `predict`. The confusion matrix and classification report are generated to evaluate precision, recall, F1-score, and support for each class.



Classification Report:

	precision	recall	f1-score	support
0	1.00	0.92	0.96	12
1	0.93	1.00	0.96	13
accuracy			0.96	25
macro avg	0.96	0.96	0.96	25
weighted avg	0.96	0.96	0.96	25

Figure 1: Confusion Matrix showing correct and incorrect classifications.

Multi-class Evaluation and Additional Metrics

For multi-class tasks, F1 scores are calculated using different averaging techniques like macro, micro, and weighted averages.
- `macro` gives equal weight to all classes
- `micro` considers total true positives/negatives
- `weighted` takes class imbalance into account

Conclusion

This analysis demonstrated model evaluation using confusion matrix, ROC curve, and precision-recall metrics. Each method provides different insights about model performance, especially in multi-class or imbalanced classification tasks. Using these evaluation tools, we can optimize hyperparameters, compare models, and make data-driven improvements.