**67. Python ROC Curves in Warehouse Logistics Analysis**

Today, I'll cover a topic in the lab that, while not specifically related to support vector classifiers, is essential for evaluating various types of models: **ROC curves**, or Receiver Operator Characteristic curves. In the context of warehouse logistics, these curves are invaluable for summarizing the performance of classifiers used to predict outcomes such as inventory shortages, order fulfillment efficiency, or equipment failure rates at different threshold levels.

To recap, in Chapter 4, I explored classifiers that predict the probability of an event, such as a potential stockout or a delayed shipment. By adjusting the decision threshold for these predictions, different outcomes are generated—each with its unique accuracy, false-positive rate, and so forth. An **ROC curve** effectively summarizes this accuracy as a function of the threshold, providing a holistic view of a classifier's performance.

To create an ROC curve in Python, I use the ROCCurveDisplay function from the scikit-learn metrics package and the from\_estimator method. By providing it with a classifier that is a scikit-learn estimator, the function can vary the threshold to plot the accuracy curve as the threshold parameter changes. Note that ROC curves are mainly used for binary outcomes. For example, in warehouse logistics, this could involve a classifier predicting whether or not an item will run out of stock within a certain period.

**Evaluating the Classifier on Training and Test Data**

I start by using the tuned Support Vector Machine (SVM) model I discussed earlier to evaluate the **ROC curve** on both training and test data. First, I generate the ROC curve for the training data. When I evaluate the model on test data, I expect the performance to drop slightly, as is often the case when moving from training to test datasets.

The ROC curve plot consists of two axes: the **true-positive rate (TPR)** on the y-axis and the **false-positive rate (FPR)** on the x-axis. In warehouse logistics terms, the true-positive rate could represent correctly predicted stockouts (actual shortages correctly identified), while the false-positive rate might represent false alarms (predicting a shortage when there isn't one). A good classifier will appear high in the upper-left corner of the plot, with a high TPR and a low FPR.

Since the first ROC curve is plotted using training data, it performs very well, showing a nearly perfect model fit. The **Area Under the Curve (AUC)** is a summary measure of the ROC curve, which in this case is about 99%. This high AUC indicates that the model is close to perfect; for comparison, a random guessing model would achieve an AUC of about 50%. A perfect model would have an AUC of 100%.

**Comparing Training and Test Data Performance**

Next, I create several ROC plots to compare the performance on training and test data. Here, I've adjusted the classifier to make it more flexible. The model, when fit to training data, achieves 100% accuracy, which is expected for a highly flexible model on training data. However, it's essential to see how it performs on unseen test data to understand its true predictive power in a real-world warehouse logistics setting.

Now, I plot the ROC curve for the test data and overlay it onto the same plot as the training data using the **axis** argument. Many Python functions that generate plots (like those in scikit-learn and pandas) allow the use of the axis argument to add more elements to an existing plot. By redisplaying the figure, I can see both curves simultaneously.

In this case, the red curve represents the training data, while the blue curve represents the test data. As anticipated, the blue curve does slightly worse than the red one, reflecting the model's performance on test data. However, it still performs well, with an AUC of about 90%, indicating a robust model that generalizes well to unseen data.

**Conclusion**

This overview wraps up the discussion for today. While ROC curves are not unique to any specific type of classifier, they are an essential tool for data analysis in warehouse logistics. They help summarize and visualize the performance of predictive models—whether predicting potential inventory shortages, delays in order fulfillment, or machine failures.

For those interested in extending their skills further, I recommend exploring the lab exercise on support vector machines with multiple classes, which can provide additional insights into handling more complex classification problems, such as segmenting products into high, medium, or low risk of stockouts based on various logistic factors. This exercise can be done offline and will be valuable in refining predictive modeling strategies in warehouse management.