**28. Python K-Nearest Neighbors (KNN) for Predicting Shipping Outcomes in a Warehouse**

The final classification method I will discuss in today’s project is **K-Nearest Neighbors (KNN)**. Although KNN is a different classifier from Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA), the way I use it with scikit-learn is quite similar. The primary difference is in the specific classifier class used from scikit-learn—in this case, it is the KNeighborsClassifier. I start by constructing a KNN classifier using one nearest neighbor. From there, the remaining code for fitting the model to the training data and predicting on the test data is identical to the previous classifiers. I then compute the confusion matrix to evaluate performance.

Using one nearest neighbor makes the KNN classifier highly flexible, but this can also lead to high variance and potential overfitting, particularly with small datasets. Indeed, the accuracy of the classifier with one nearest neighbor is around 50%, which is as bad as random guessing. This result suggests that the model is not very good in this configuration.

To see if I can improve the performance, I increase the number of neighbors to three. With three neighbors, the accuracy improves to about 53%, which is better but still not as good as the discriminant analysis methods (LDA and QDA) I explored earlier. This result highlights the adaptability and flexibility of K-Nearest Neighbors, but also its susceptibility to overfitting, especially on smaller datasets.

To make selecting the optimal number of neighbors easier, I introduce a new dataset to further explore KNN tuning. I use the **Caravan** dataset from the ISLP package, which deals with predicting whether people will purchase caravan insurance or not. This dataset presents a highly imbalanced classification problem, with only 6% of the observations corresponding to a purchase. Unlike the balanced stock market data used earlier, this imbalance requires careful handling. By the way, a "caravan" refers to a campervan in regions like Australia, New Zealand, and South Africa, and an RV in North America.

For this dataset, I use all features except the Purchase column to create the feature set. Unlike the previous dataset, which had straightforward lagged features (Lag1, Lag2), the Caravan dataset contains many features measured in different units. Because KNN relies on calculating distances between points, any feature with significantly different units can dominate the distance calculation. To address this, I standardize the features before applying the KNN classifier. For this, I use the StandardScaler from scikit-learn, which is an example of a transformer. Transformers like these are used to derive new features from existing ones by scaling them to have a mean of zero and a variance of one.

Once the features are standardized, I split the dataset into training and test sets. I then apply the K-Nearest Neighbors classifier with various numbers of neighbors to determine the optimal choice. This process is similar to the stock market K-Nearest Neighbors example but with different data. Using one nearest neighbor, I achieve about 11% accuracy, which is better than the 6% I would get if I used only the intercept in the model. Although there is some improvement, it is not a dramatic one.

Choosing the number of nearest neighbors involves selecting a tuning parameter for the estimator, a common task in machine learning. To find the best number of neighbors, I loop over a range from 1 to 5 neighbors. Inside the loop, I fit the K-Nearest Neighbors classifier on the training data, predict on the test data, and compute the confusion matrix. I then calculate the accuracy and print it out as a function of the number of nearest neighbors. Later, I can use plots to visualize the accuracy or mean squared error as a function of the tuning parameter, but for now, a simple printout suffices.

Interestingly, for the test and training splits, most numbers of neighbors achieve around 14% or 15% accuracy, except for the configuration with four neighbors, which shows an anomaly. It only predicted "Forward Rent," leading to a notably different result compared to others. This behavior is a reminder of the noisy nature of the K-Nearest Neighbors classifier.

That concludes today’s project discussion on K-Nearest Neighbors. I encourage you to explore further sections, such as Poisson regression, which will be covered offline. Up next, I’ll delve into the concept of **Cross-Validation**, a critical method for assessing the performance and robustness of machine learning models.