(1)What’s the difference between the different SVM kernels?

SVM kernels are functions that transform data into a new, higher-dimensional space. This transformation makes it easier to find a clear boundary (a hyperplane) to separate the different classes.

Linear Kernel: The simplest kernel, it's used for data that can be separated by a straight line. It's fast and works well when the number of features is very high.

Polynomial Kernel: This kernel finds a curved decision boundary by transforming data into a polynomial space. Use it when data has a non-linear relationship.

RBF Kernel: The most versatile and powerful kernel, it maps data to an infinite-dimensional space to find a complex, non-linear boundary. It's a great default option when linear models fail and the data's relationship is unknown.

(2) When would you use each one, and how do their outputs differ?

(a)Linear Kernel

Performance: Recall for fraud was 0.88 (very good at catching fraud), but precision was only 0.33 (many false alarms). AUC was 0.9515.

When to use: Suitable when the data is close to linearly separable or when a fast and interpretable model is needed.

Output difference: Creates a straight-line (hyperplane) boundary. Focuses on catching almost all fraud (high recall) but misclassifies many normal transactions as fraud.

(b) Polynomial Kernel

Performance: Recall = 0.80, precision = 0.66, and AUC = 0.9212. More balanced between precision and recall compared to linear, but overall AUC is lower.

When to use: Useful when fraud patterns follow polynomial-like interactions (squares, cubes, etc.) between features.

Output difference: Decision boundary is curved (polynomial shape). Reduces false alarms compared to linear, but doesn't separate classes as strongly as RBF.

(c) RBF Kernel

Performance: Recall = 0.77, precision = 0.54, and AUC = 0.9723 (best overall). Slightly lower recall than linear, but much higher AUC and better balance between false positives and false negatives.

When to use: Serves as the default kernel for most real-world problems, especially when relationships are complex and nonlinear (like fraud detection).

Output difference: Produces flexible, nonlinear boundaries that can wrap around clusters of fraud cases. Provides the best overall separation in the dataset.