1. An SVM classifier can provide confidence scores using decision values (distance from the hyperplane), where larger magnitudes indicate higher confidence. To obtain probabilities, methods like Platt Scaling fit a sigmoid function to these scores, mapping them to a [0, 1] range. However, this requires additional computation and is less robust compared to probabilistic models like Logistic Regression.
2. If an SVM with an RBF kernel is underfitting, increase gamma to make the decision boundary more responsive to individual data points and C to prioritize reducing training errors over maximizing the margin. Both adjustments reduce underfitting, but tuning must be done carefully (e.g., with grid search or cross-validation) to avoid overfitting.
3. A model is **ϵ-insensitive** when it ignores small errors within a range $[-\epsilon,+\epsilon]$ (tolerance zone) and only penalizes significant deviations (outside this range).We define a Define an ϵ-Insensitive Loss Function: 0 if $|y - f(x)| <= \epsilon$, $|y - f(x)|$otherwise.(e.g. let $\epsilon = 50$. if the actual value $y = 200$ and predicted value $f(x) = 198$ the error is $|y - f(x)| = 2$ which is within $[-\epsilon,+\epsilon] = [-50, +50]$ so the loss is zero. if the actual value $y = 200$ and predicted value $f(x) = 260$ the error is $|y - f(x)| = 60$ so the loss is $60 - 50 = 10$.
4. ROC curve: Plots the **True Positive Rate (TPR)** vs. **False Positive Rate (FPR)** across various thresholds. A model with a better discriminative ability will have an ROC curve that lies closer to the top-left corner (high TPR and low FPR).
   AUC: The **AUC** is the area under the **ROC curve** and is a single scalar value that summarizes the entire ROC curve. AUC = 1 means a perfect classifier. AUC = 0.5 No discriminative ability (random guesses). AUC < 0.5. its Commonly used in binary classification.

F1-score: The **F1-score** is a single metric combining **precision** and **recall** into a harmonic mean. This balances the two metrics, particularly when classes are imbalanced.
Precision = TP/(TP + FP) (Measures how many of the predicted positives are actually positive.)
Recall = TP/(TP + FN) (Measures how many of the actual positive instances are identified.)
F1-Score = 2 * (Precision * Recall)/(Precision + Recall)
The F1-score is between **0 and 1. 1 means Perfect precision and recall and 0 means Either precision, recall, or both are zero.**
It balances the trade-off between false positives and false negatives.

5. The **threshold value** in classification is the decision boundary that determines whether a prediction will be classified as one class (e.g., 1) or the other (e.g., 0). The most common default value is **0.5**, meaning if the predicted probability is greater than 0.5, the observation is assigned to class 1, otherwise, it is assigned to class 0. Adjusting this threshold impacts the balance between **false positives**, **false negatives**, **precision**, and **recall**, leading to a trade-off between these metrics.
   Lower Thresholds (e.g., 0.3) : The model is more likely to classify observations as **positive**, reducing **false negatives**. However, this leads to more **false positives** because more observations are being classified as 1.
   Higher Thresholds (e.g., 0.3) : This reduces **false positives**, but increases **false negatives** because fewer observations are being classified as 1.

The **trade-off between false positives and false negatives** arises because increasing one typically leads to an decrease in the other when making decisions based on a classification model's probability threshold.

According to the formulas:
High Threshold: Reduces false positives but increases false negatives this leads to higher precision.
Low Threshold: Reduces false negatives but increases false positives this leads to higher recall.

6. RBF Kernel.ipynb
   Low lambda : the kernel's spread is **wide**. Distant points are considered similar. Distant points still have relatively high similarity values because the effect of the Euclidean distance is dampened over a large range.
   High lambda : the kernel's spread is **narrow**. Only very nearby points are considered similar. Distant points have very low similarity because the kernel decays rapidly with distance.
7. Precision: Precision tells you how many of the predicted positive cases were actually positive. High precision means the model has few false positives.
   Precision is crucial when the cost of false positives is high
   TP / (TP + FP)
   Recall: Recall tells you how many of the actual positive cases were correctly identified by the model. High recall means the model has few false negatives. Recall is important when the cost of missing a positive instance is high
   TP / (TP + FN)

F1-Score: The F1-score combines both precision and recall into a single metric. It is particularly useful when there is an uneven class distribution.
The F1-score is helpful when there is a need to balance false positives and false negatives. A high F1-score means the model has both high precision and high recall, performing well in both respects.

8. Min-Max Scaling: Min-Max Scaling transforms features to a fixed range, typically [0, 1]. It maintains the relationship between values but scales them proportionally to fall within the desired range.
   Formula: $X' = (X - X_{min}) / (X_{max} - X_{min})$
   X' is scaled value
   Min-Max Scaling is suitable when features need to be scaled to a fixed range, such as for machine learning algorithms sensitive to magnitude differences
   This technique can be heavily impacted by outliers because outliers can shift Xmin or Xmax and compress other data ranges.
   Z-score Normalization: Z-score normalization scales data based on its mean and standard deviation. It transforms the data into a distribution with a mean of 0 and a standard deviation of 1. Unlike Min-Max Scaling, Z-score normalization does not bound values to a fixed range.
   Formula: $X' = (X - \mu) / \sigma$
   Suitable for algorithms like Support Vector Machines (SVMs), Principal Component Analysis (PCA), and linear regression.
   Z-score normalization is less affected by outliers compared to Min-Max scaling because it uses statistical properties rather than absolute value ranges.

Good when scaling data to specific ranges isn't necessary but centering it around a mean is important.
Robust Scaling: Robust Scaling uses statistics that are robust to outliers (median and interquartile range) to scale data. This is a good choice when the data contains outliers that may distort scaling if using Min-Max or Z-score normalization.
Formula: $X' = (X - \text{median}(X)) / IQR$
IQR = interquartile range, calculated as $Q3 - Q1$
Robust scaling is effective when your dataset has outliers since it uses the median and IQR instead of the mean and standard deviation. When preparing data for models like tree-based methods or models that can be affected by extreme values, robust scaling is beneficial.

9. Nested Cross-Validation: Nested Cross-Validation is a statistical validation technique used to evaluate the performance of machine learning models while simultaneously performing hyperparameter tuning. It involves two levels of cross-validation:
   Outer Loop: Divides the data into training and testing sets.
   Inner Loop: Performs hyperparameter tuning on the training set from the outer loop.
   The performance of the model is then evaluated on the held-out *test* data from the outer loop.
   We use Nested Cross-Validation when **hyperparameter tuning** is part of the model training process.
   Prevent **data leakage** by separating the data used for selecting hyperparameters from the test set.
   Especially beneficial when the dataset is small, and a single cross-validation could lead to overfitting by tuning hyperparameters on the same data used for evaluation.

To estimate a model's **generalization error** reliably without bias from hyperparameter tuning.
5x2 Cross-Validation: The **5x2 cross-validation** is a specific type of repeated k-fold cross-validation where the dataset is split into 5 folds **2 times independently**. This allows better stability and more reliable performance estimates.
Randomly split the entire dataset into 5 folds **twice independently** (e.g., randomly shuffle the data before splitting each time). Train and evaluate the model on each of the 5 folds, while rotating the train-test splits. Average the results over both splits for improved robustness.
We use 5x2 Cross-Validation To ensure that the model is tested on varying train-test splits without overfitting to a single split.
It reduces the chance that any single random split introduces bias into the results. Compared to other forms of repeated k-folds, it is computationally efficient while still providing reliable estimates.