# Machine Learning - 2nd Assignment

## Task 2: SVM Classifier Experimentation

**Name:** [Your Name] **Sub Code:** BCS602 **Year:** 2024-2025 **Semester:** VI

## 1. Detailed Explanation of Dataset Used

### a. Dataset Name and Source

The dataset used for this task is the **Breast Cancer Wisconsin (Diagnostic) Dataset**. \* **Source:** This dataset is included in sklearn.datasets module (load\_breast\_cancer()). It is a classic and commonly used dataset for binary classification tasks.

### b. Size and Structure

* **Size:** The dataset contains **569 instances (samples)** and **30 numeric, predictive attributes (features)**.
* **Target Variable:** The target variable is binary, indicating whether a tumor is **malignant (0)** or **benign (1)**.
  + Number of malignant samples: [Count from your notebook, e.g., 212]
  + Number of benign samples: [Count from your notebook, e.g., 357]
* **Attributes:** The 30 features are real-valued characteristics computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image (e.g., radius, texture, perimeter, area, smoothness, compactness, concavity, concave points, symmetry, fractal dimension – mean, standard error, and “worst” or largest of these features).

### c. Preprocessing of Dataset

The following preprocessing steps were performed: 1. **Loading Data:** The dataset was loaded using sklearn.datasets.load\_breast\_cancer(). Features were placed in a Pandas DataFrame and the target in a Pandas Series. 2. **Train-Test Split:** The dataset was split into a training set (70%) and a testing set (30%) using train\_test\_split. random\_state=42 was used for reproducibility, and stratify=y was used to ensure that the proportion of target classes was similar in both training and testing sets. 3. **Feature Scaling:** The features in both the training and testing sets were scaled using StandardScaler from scikit-learn. This standardizes features by removing the mean and scaling to unit variance. SVMs are sensitive to feature scaling, so this step is crucial for optimal performance.

## 2. Explain the Working of Algorithm on Selected Dataset with Necessary Figures

### a. Support Vector Machine (SVM) Algorithm

Support Vector Machine (SVM) is a powerful supervised machine learning algorithm used for both classification and regression tasks. For classification, SVM aims to find an optimal hyperplane in an N-dimensional space (where N is the number of features) that distinctly classifies the data points. The optimal hyperplane is the one that has the largest margin, i.e., the maximum distance between data points of different classes (called support vectors).

**Key Concepts in SVM:** \* **Hyperplane:** A decision boundary that separates the classes. \* **Margin:** The distance between the hyperplane and the closest data points (support vectors) from either class. SVM tries to maximize this margin. \* **Support Vectors:** The data points that lie closest to the hyperplane and influence its position and orientation. \* **Kernels:** SVM can use kernel functions to transform the input data into a higher-dimensional space, allowing it to find non-linear decision boundaries. Common kernels include: \* **Linear:** For linearly separable data. \* **Polynomial (poly):** For polynomial decision boundaries. \* **Radial Basis Function (rbf):** A popular choice, can map samples to a higher-dimensional space and handle complex relationships. \* **Sigmoid:** Can also be used for non-linear classification. \* **Hyperparameters:** \* **C (Regularization Parameter):** Controls the trade-off between achieving a low training error (fitting the training data well) and a low testing error (generalizing to new data). A small C creates a wider margin but might misclassify some training points. A large C creates a narrower margin and tries to classify all training points correctly, potentially leading to overfitting. \* **Gamma (Kernel Coefficient for ‘rbf’, ‘poly’, ‘sigmoid’):** Defines how far the influence of a single training example reaches. Low gamma means a larger similarity radius, leading to a smoother boundary (points further away are considered). High gamma means a smaller similarity radius, leading to a more complex, potentially overfit boundary (only points close by are considered).

### b. Experimentation and Results

Several experiments were conducted to observe the impact of different kernels, C values, and gamma values on the SVM classifier’s performance, measured primarily by accuracy on the test set.

**Experiment 1: Different Kernels** (Default C=1.0, default gamma=‘scale’ for RBF/Poly/Sigmoid)

**Figure 1: Accuracy for Different SVM Kernels** *(Here, you will insert the bar plot image of accuracies for different kernels from your notebook. Take a screenshot, save it as svm\_kernel\_accuracy.png in your Task2\_SVM\_Classifier/report/ folder, and then reference it.)* ![SVM Kernel Accuracy Plot](./svm\_kernel\_accuracy.png)

**Observations from Experiment 1:** \* The linear kernel achieved an accuracy of approximately [Accuracy value from your notebook, e.g., 97.08%]. \* The rbf kernel achieved an accuracy of approximately [Accuracy value, e.g., 97.66%]. \* The poly kernel achieved an accuracy of approximately [Accuracy value, e.g., 94.15%]. \* The sigmoid kernel achieved an accuracy of approximately [Accuracy value, e.g., 94.74%]. *(Adjust these based on your actual notebook output. Also, briefly comment on precision/recall for benign/malignant if notable differences occurred.)*

**Experiment 2: Different C values (using ‘rbf’ kernel, gamma=‘scale’)**

**Figure 2: Accuracy for Different C Values (RBF Kernel)** *(Insert the line plot image of accuracies for different C values. Save as svm\_c\_value\_accuracy.png.)* ![SVM C Value Accuracy Plot](./svm\_c\_value\_accuracy.png)

**Observations from Experiment 2:** \* With the RBF kernel and gamma=‘scale’, varying the C parameter showed the following trend: \* C = 0.01: Accuracy ~ [Value]% \* C = 0.1: Accuracy ~ [Value]% \* C = 1.0: Accuracy ~ [Value]% \* C = 10: Accuracy ~ [Value]% \* C = 100: Accuracy ~ [Value]% \* A C value of [Optimal C from this experiment, e.g., 1 or 10] appeared to provide the best performance in this specific experiment. Very low C values might lead to underfitting, while very high C values increase the risk of overfitting.

**Experiment 3: Different Gamma values (using ‘rbf’ kernel, C=[Your chosen C from Exp2, e.g., 1.0])**

**Figure 3: Accuracy for Different Gamma Values (RBF Kernel, C=[Your C])** *(Insert the line plot image of accuracies for different gamma values. Save as svm\_gamma\_value\_accuracy.png.)* ![SVM Gamma Value Accuracy Plot](./svm\_gamma\_value\_accuracy.png)

**Observations from Experiment 3:** \* With the RBF kernel and C=[Your C], varying the gamma parameter showed: \* gamma = 0.001: Accuracy ~ [Value]% \* gamma = 0.01: Accuracy ~ [Value]% \* gamma = 0.1: Accuracy ~ [Value]% \* gamma = 1: Accuracy ~ [Value]% \* gamma = ‘auto’: Accuracy ~ [Value]% \* A gamma value of [Optimal gamma from this experiment, e.g., 0.01 or ‘scale’ if it was tested and did well] seemed optimal. Small gamma values create smoother decision boundaries, while large values can lead to overfitting by making the model too sensitive to individual data points.

**Experiment 4: GridSearchCV for Optimal Hyperparameters** A GridSearchCV was performed to systematically search for the best combination of C, gamma, and kernel (tested ‘rbf’ and ‘linear’). \* **Best Parameters Found:** \* Kernel: [Value from grid\_search.best\_params\_['kernel']] \* C: [Value from grid\_search.best\_params\_['C']] \* Gamma: [Value from grid\_search.best\_params\_['gamma'] if applicable, else 'N/A'] \* **Best Cross-Validation Accuracy:** Approximately [Value from grid\_search.best\_score\_]% \* **Test Set Accuracy with Best Model:** Approximately [Value from accuracy\_best]%

*(You can also include the classification report for the best model from GridSearchCV here if you wish, or a screenshot of its confusion matrix).*

## 3. Interpretation of Tasks and Results

The experiments conducted on the Breast Cancer dataset using the SVM classifier demonstrated the significant impact of hyperparameter selection on model performance.

* **Kernel Choice:** The linear and rbf kernels generally outperformed the poly and sigmoid kernels for this dataset. The RBF kernel’s ability to handle non-linear relationships, coupled with proper tuning of C and gamma, often yields high accuracy. The linear kernel is simpler and faster, performing well when the data is largely linearly separable or when the number of features is high.
* **C (Regularization):** The C parameter balances the margin width and training error. A moderately chosen C (e.g., 1 to 10 for this dataset with RBF kernel) typically provided good generalization. Too small a C might underfit (oversimplify), while too large a C can overfit (memorize training data, perform poorly on unseen data).
* **Gamma (Kernel Coefficient):** For the RBF kernel, gamma dictates the influence of individual training samples. An appropriate gamma value (e.g., scale or values around 0.01 for this dataset) is crucial. If gamma is too large, the decision boundary becomes too complex and overfits. If too small, the model becomes too constrained and underfits.
* **Optimal Configuration:** Based on the GridSearchCV results (and manual experiments), the most effective SVM configuration for this dataset was found to be a kernel of [best\_kernel from GridSearchCV], with C=[best\_C] and gamma=[best\_gamma if applicable], achieving a test accuracy of approximately **[best\_accuracy from GridSearchCV]%**. This configuration provided a robust model capable of accurately distinguishing between malignant and benign tumors.

**Conclusion for Task 2:** This task successfully demonstrated how to experiment with different SVM hyperparameters (kernels, C, gamma) and evaluate their effect on a classification problem. The Breast Cancer dataset was effectively classified, with the best-tuned SVM models achieving high accuracy. The process highlighted the importance of feature scaling and systematic hyperparameter tuning (e.g., using GridSearchCV) for optimizing SVM performance.