Name: Sai Ram Gunturu
ID: C00313478

Support Vector Machine (SVM) - Iris Classification

Data Analytics & Algorithms Log

- **Algorithm Used:** Support Vector Machine (SVM)
- **Dataset Used:** Iris Dataset
- **Framework:** CRISP-DM

Original Workbook Link: [Notebook]

Phases	Changes Made	Reason for the change	Duration	Difficulty level (1-10)
Datasets selection	Changed the synthetic dataset to real time iris dataset	I thought of implementing it on some real-world datasets to deepen my understanding of the model.	10 min	3
Algorithm Version	Changed from LinearSVC to multiple kernels (linear, poly, sigmoid and rbf)	Since versicolor and virginica aren't perfectly linearly separable, I thought to try all kernels to select the best kernel to move forward	30 min	6
Data Preprocessing	StandardScalar	Standardized the data for better prediction	30 min	6
Kernels used and their results	Linear, Poly, RBF and Sigmoid	Cross validated different kernels to select the best kernel to move forward.	2 hours	7
Further Enhancements	GridSearchCV	After kernel results, the linear kernel outperformed all other kernels. With the linear kernel I tried fine tuning the hyperparameters to enhance the model performance.	1 hour	7

Conclusions		A Drastic decrease in overall accuracy in the enhanced model made it possible to declare that the linear kernel model with default parameters performed the best. Even though Iris dataset is a well-known linearly separable dataset, exploring different kernels, techniques for improving the performance of the model helped me to deepen my problem-solving skills, decision making skills (selecting the best kernel to improve the model).	30 min	6

1. Business Understanding

Objective:

This study aims to classify **Iris flower species** using **Support Vector Machine (SVM)** and analyze how the choice of a kernel impacts the performance of the model. The dataset consists of **three classes** (**“Setosa”, “Versicolor”, “Virginica”**), that require a **multiclass classification approach**.

Challenges Faced:

- **Selecting the appropriate kernel:** SVM supports different kernels (**Linear, RBF, Polynomial, Sigmoid**). Choosing the optimum one was important to achieve high accuracy.
- **Hyperparameter tuning:** The **C** parameter for the SVM model must be **fine-tuned** to achieve optimal performance.
- **Understanding decision boundaries:** A key part of understanding the mode was Visualizing how different kernels separate the classes.

2. Data Understanding

Dataset Overview:

- **Total Samples:** 150
- **Features Used:** “Sepal Length”, “Sepal Width” (Used the first two features for Visualisation)

- **Target Classes:**
  - Setosa (0)
  - Versicolor (1)
  - Virginica (2)
- **Class Distribution:** balanced (50 samples in each class)
- **No missing values**

Key Insights:

- **No oversampling or resampling techniques** were required as the dataset is **balanced**.
- Few need to scale the features for SVM to perform correctly.

3. Data Preparation

Preprocessing Steps:

- **Feature Scaling:** Leveraging **StandardScaler** to normalize feature values.
- **Train-Test Split:**
  - **80% Training Set (120 samples)**
  - **20% Testing Set (30 samples)**

4. Training SVM with Different Kernels

Kernels Used & Performance Details:

Kernel	Accuracy	Best For
Linear	**90%**	Best for linearly separable data
RBF	83%	Works well for non-linear relationships
Polynomial	63%	Overfits on this dataset
Sigmoid	83%	Like RBF but less stable

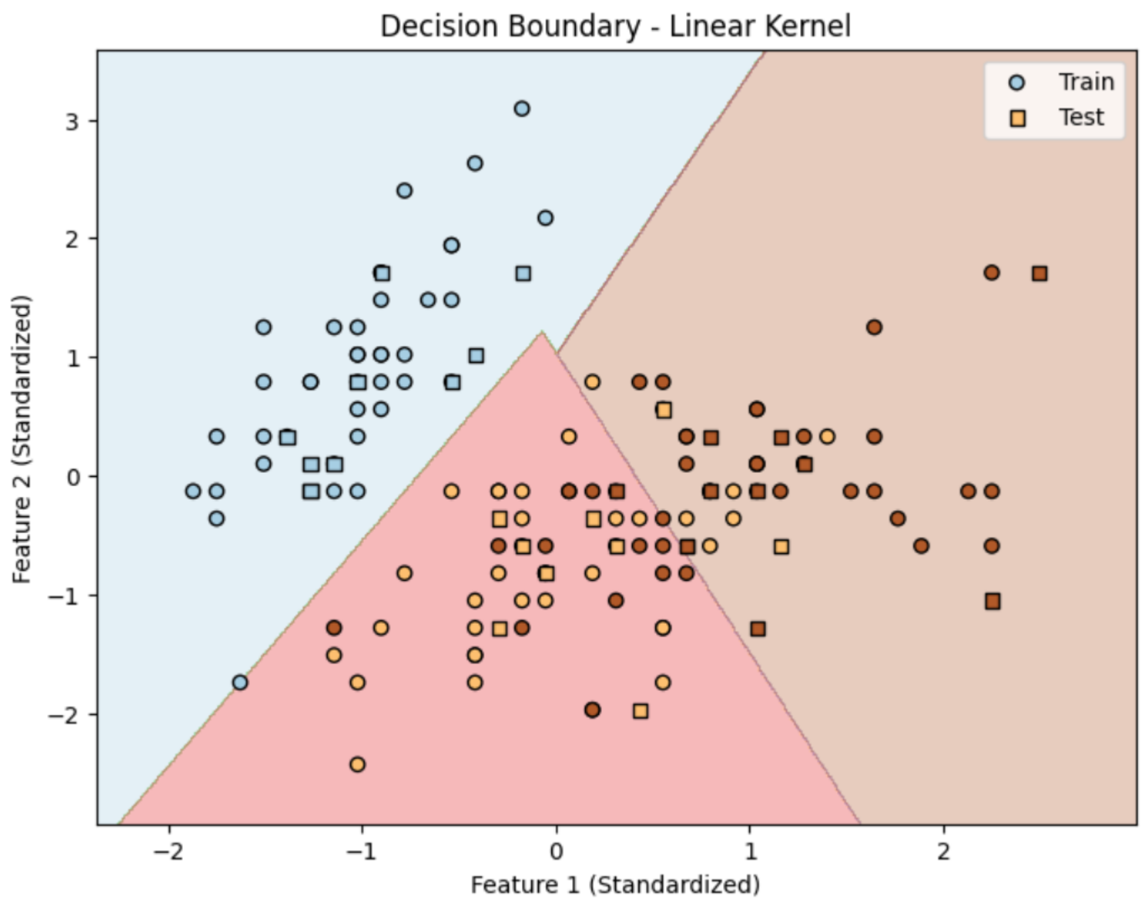
Key Observations:

- **The Linear Kernel performed the best (90% accuracy), made it the best choice for this dataset.**
- **Polynomial Kernel, overfitted**, which led to poor performance.

- **RBF and Sigmoid performed similarly but underperformed Linear.**

5. Plotting Decision Boundaries

- Plotted decision boundaries for all kernels to visualize how well they separate the classes.
- Clean class separations done by Linear Kernel proves its effectiveness.
- However Polynomial Kernel had a complex boundary that probably resulted in poor performance.



Why is this important?

Knowing how decision boundaries work in detail helps in selecting the right kernel and using hyperparameters properly to generalize better.

6. Hyperparameter Tuning (Linear SVM)

Since **Linear Kernel performed best**, we tuned it using **GridSearchCV** to fit its C **parameter**.

Results:

- **Best C Parameter:** 0.0316
- **Post Tuning Accuracy: 80%** (Lower than before 90%)
- **Performance Dropped:** Because of Lower **C** value it led to a **softer margin**, thus permitting more misclassifications.

Key Insight:

- **The SVM tuned model performed less than default SVM settings.**
- **Set C at the default value 1.0 was the good option for this dataset.**

7. Final Model Selection

Final Model: Linear SVM with Default Parameters

- Attained **90% accuracy** and surpassed all other models.
- Easy and impactful for this dataset.
- Our choice was reinforced since further hyperparameter tuning **did not improve performance**.

8. Final Conclusion

- Linear SVM was the appropriate algorithm for this dataset.
- The default model without hyperparameter tuning performed the best.
- Linear SVM's effectiveness confirmed by Decision boundary Visualisation.
- The dataset was balanced and hence there was no need of any resampling techniques.

What I Learned:

- SVM does well as long as **an appropriate kernel is selected** for the dataset.
- Linear SVM is **best for linearly separable data**.
- The (**Polynomial Kernel**) model is over-complicated which leads to **overfitting**.
- **In some cases, default hyperparameters be more effective than fine-tuning.**