# Exercise 2: Visual Exploratory Data Analysis

# Objective

To understand the role of visual exploratory data analysis (EDA) in identifying class separability and feature importance using paired plots, box plots, violin plots, and correlation heatmaps.

# Introduction

Visual EDA helps in understanding data distribution, relationships, and class separability. Before applying machine learning models, visualizations help identify:  
- Which features best separate the classes.  
- Which features have high within-class variability or overlap.  
- Correlation between features to avoid redundancy.  
  
Visualizations used:  
1. Pair Plots: Show pairwise relationships and class clusters.  
2. Box Plots: Display spread (IQR) and outliers.  
3. Violin Plots: Show distribution density and spread.  
4. Correlation Heatmaps: Summarize linear relationships between features.

# Dataset Used

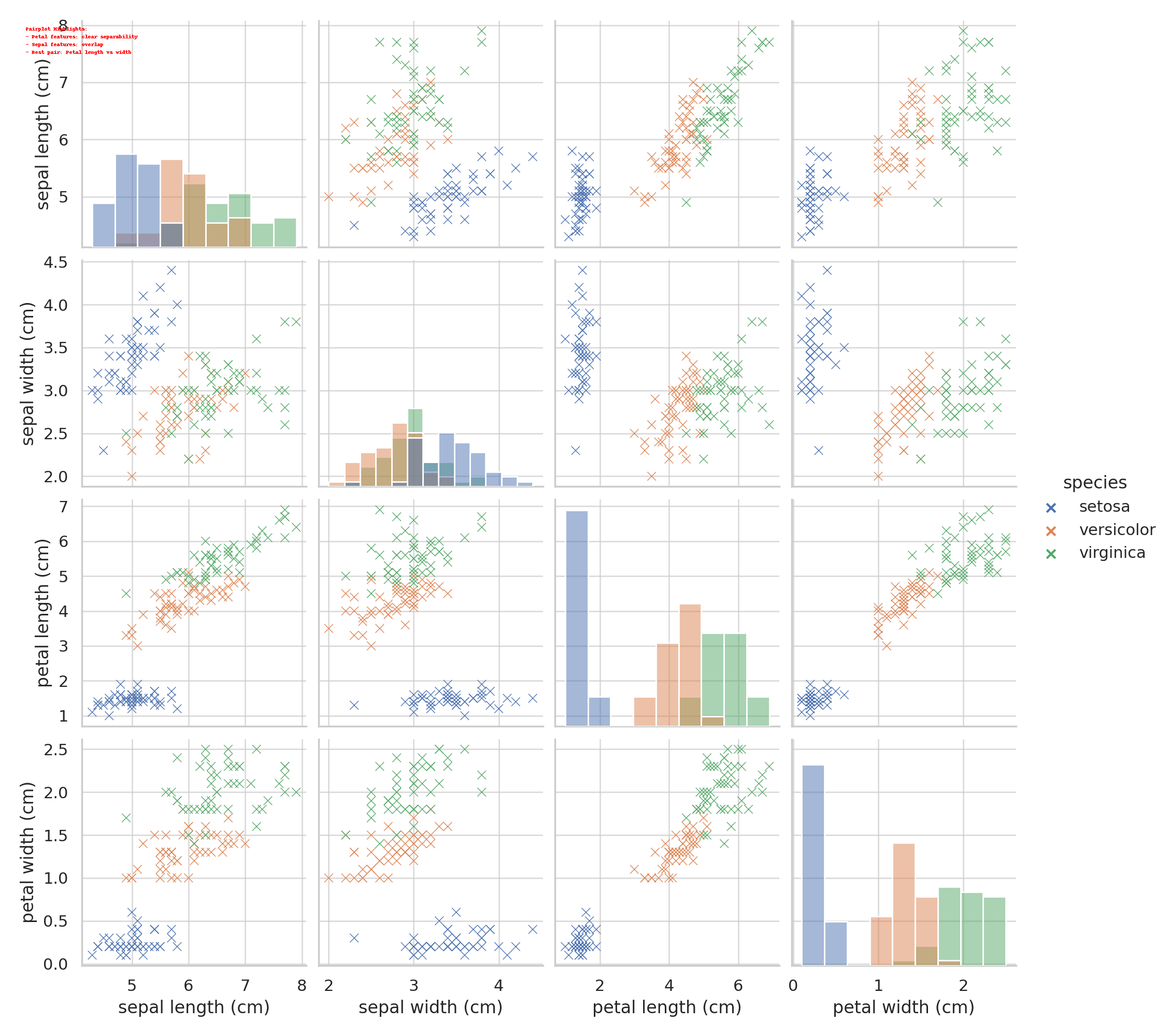
Dataset: Iris Dataset  
Features: Sepal length (cm), Sepal width (cm), Petal length (cm), Petal width (cm)  
Target: Species (Setosa, Versicolor, Virginica)

# Python Program

# Load Dataset  
from sklearn.datasets import load\_iris  
import pandas as pd  
import seaborn as sns  
import matplotlib.pyplot as plt  
  
iris = load\_iris()  
df = pd.DataFrame(iris.data, columns=iris.feature\_names)  
df["species"] = pd.Categorical.from\_codes(iris.target, iris.target\_names)  
  
# Pair Plot  
sns.pairplot(df, hue="species", diag\_kind="hist")  
plt.show()  
  
# Box Plots  
for col in iris.feature\_names:  
 sns.boxplot(x="species", y=col, data=df)  
 plt.title(f"Boxplot: {col}")  
 plt.show()  
  
# Violin Plots  
for col in iris.feature\_names:  
 sns.violinplot(x="species", y=col, data=df)  
 plt.title(f"Violin Plot: {col}")  
 plt.show()  
  
# Correlation Heatmap  
sns.heatmap(df.corr(numeric\_only=True), annot=True, cmap="coolwarm")  
plt.title("Correlation Heatmap")  
plt.show()

# Annotated Visualizations

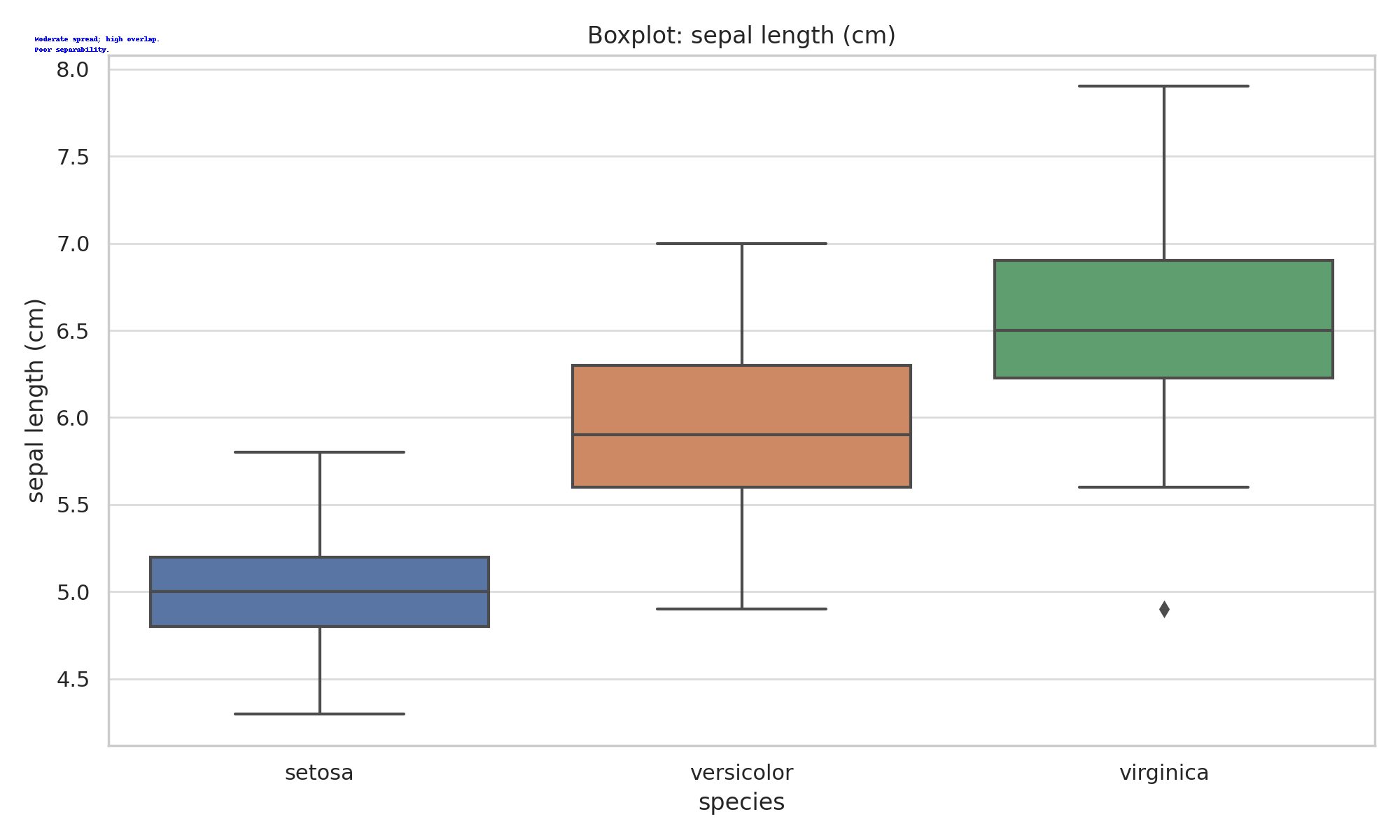
## Pair Plot (Annotated)



Highlights: Petal features show clear class separation; Sepal features overlap. Best pair: Petal length vs Petal width.

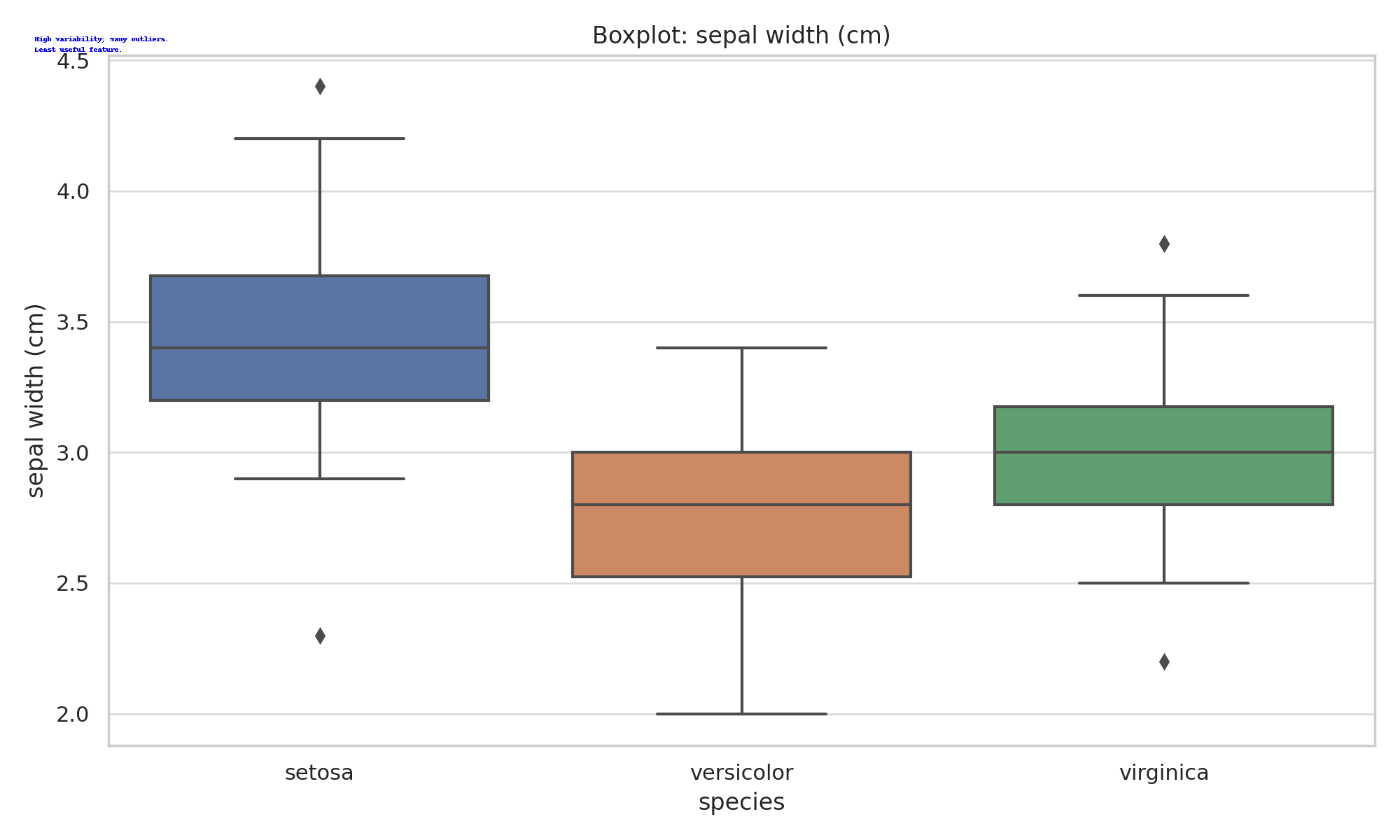
# Box Plot Analysis (Annotated)

## Sepal Length (Cm)



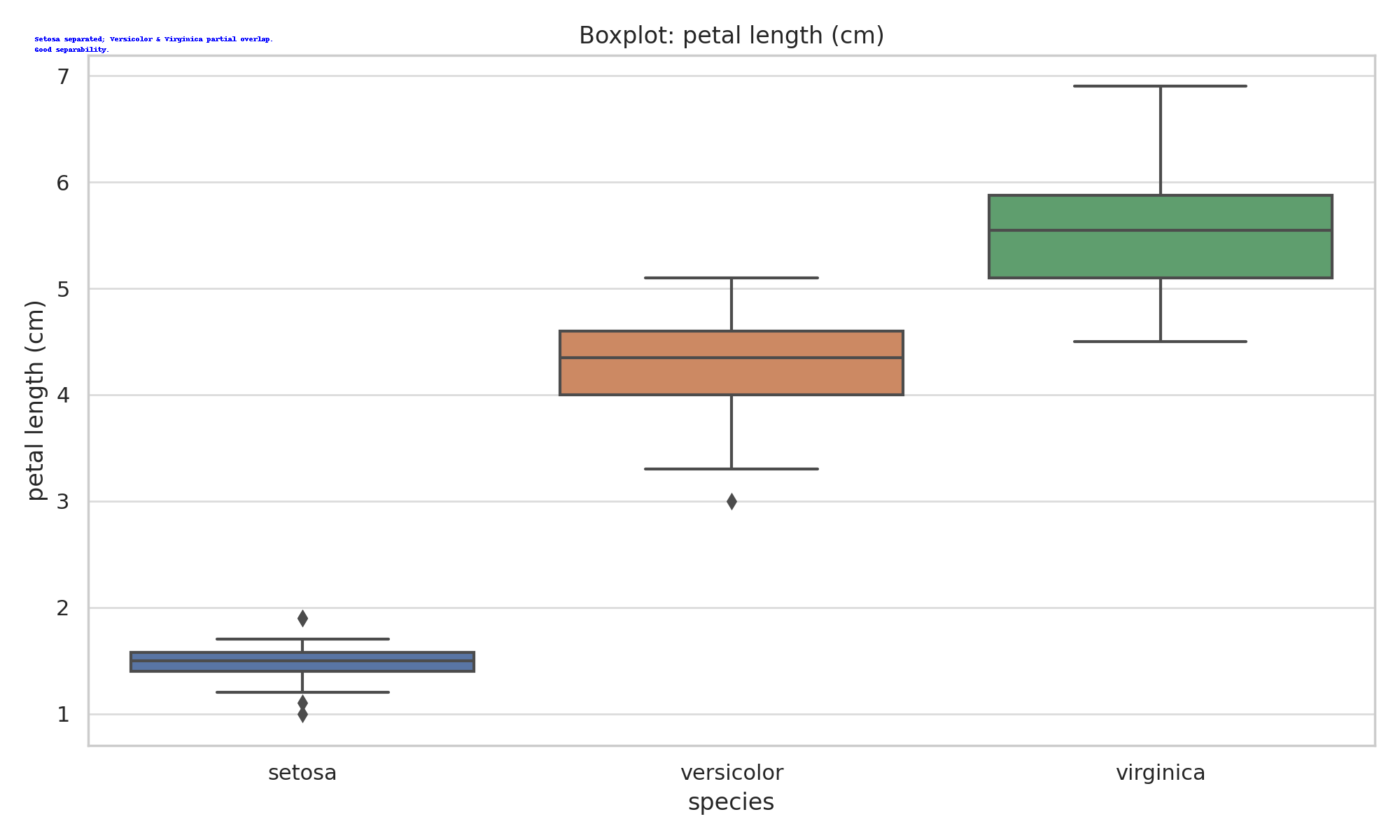
Moderate spread; high overlap.  
Poor separability.

## Sepal Width (Cm)



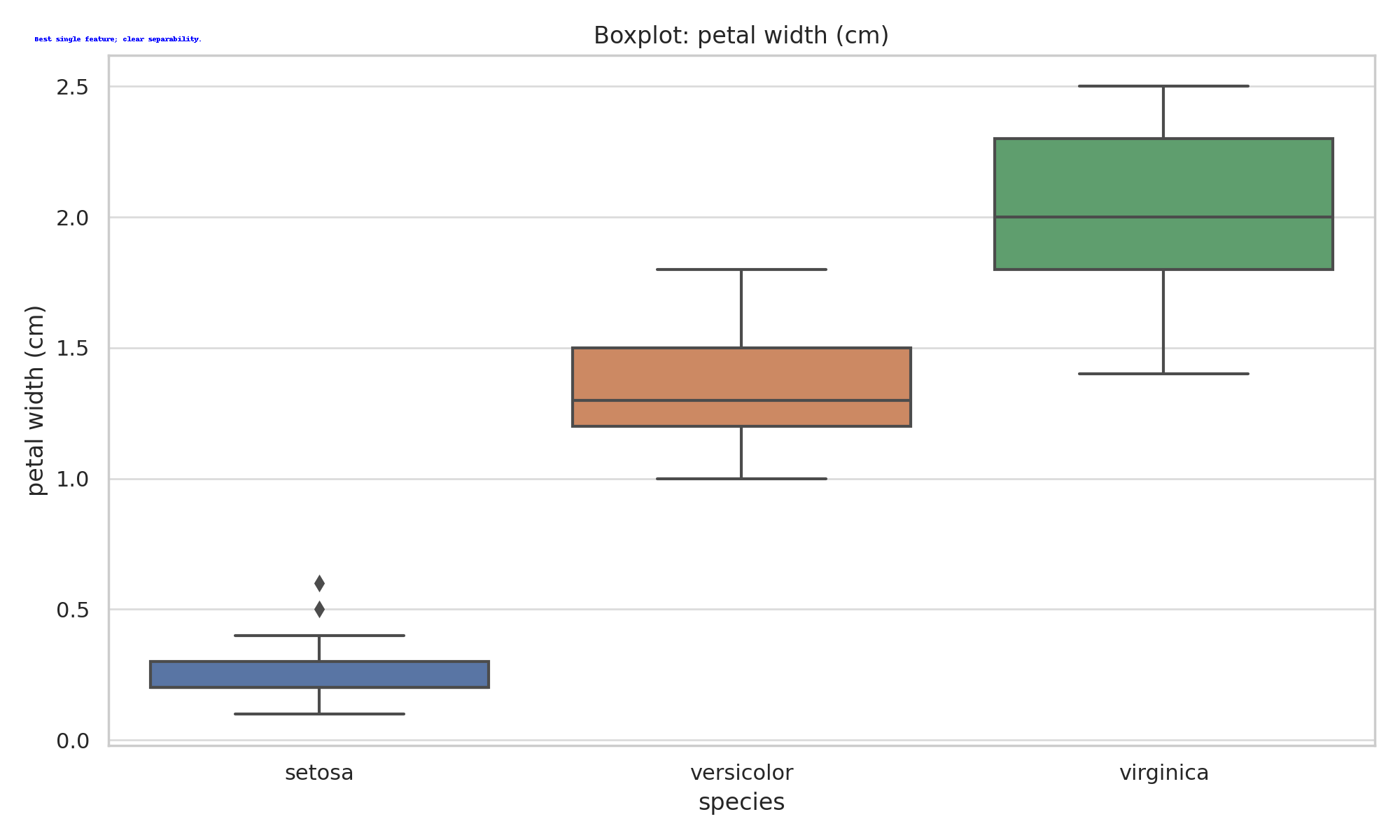
High variability; many outliers.  
Least useful feature.

## Petal Length (Cm)



Setosa separated; Versicolor & Virginica partial overlap.  
Good separability.

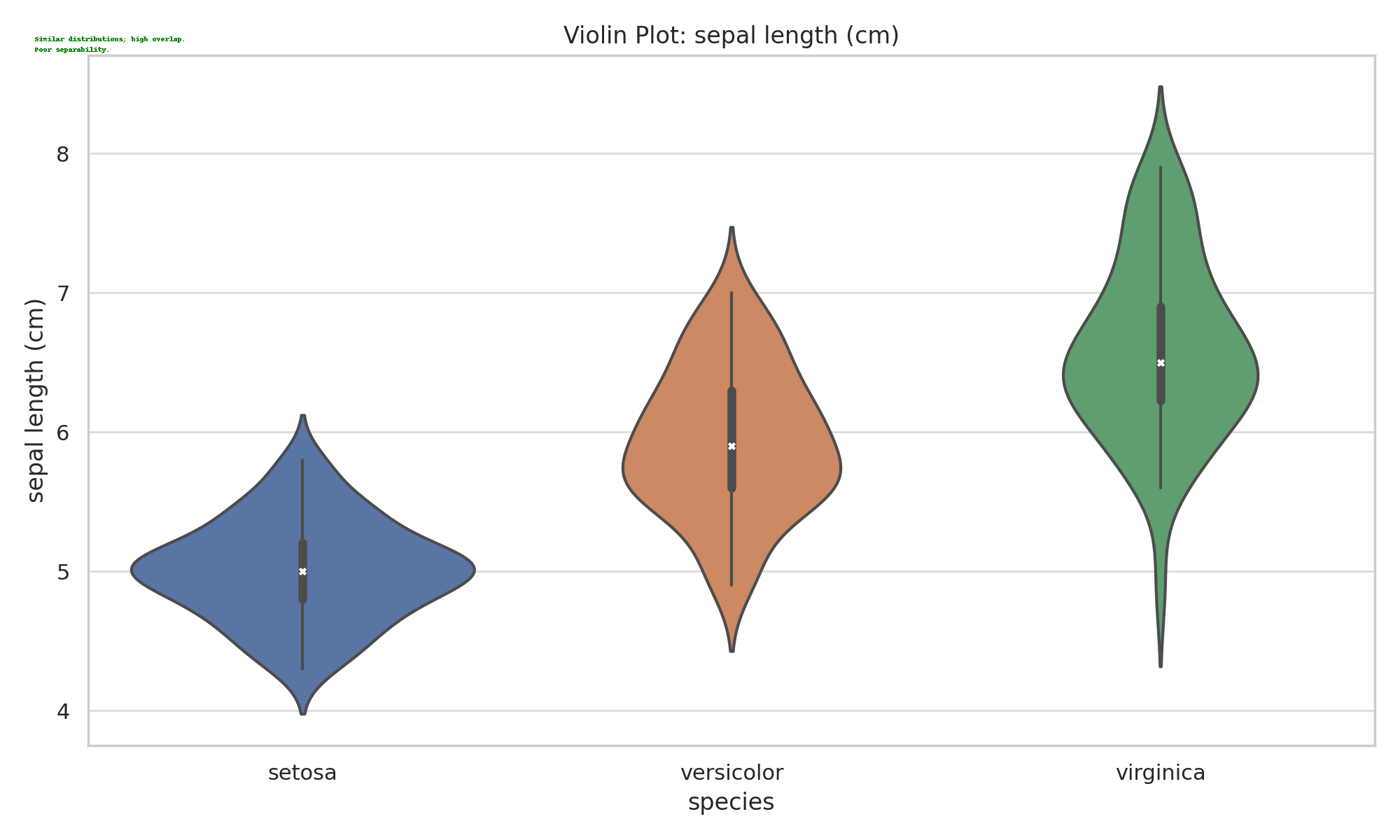
## Petal Width (Cm)



Best single feature; clear separability.

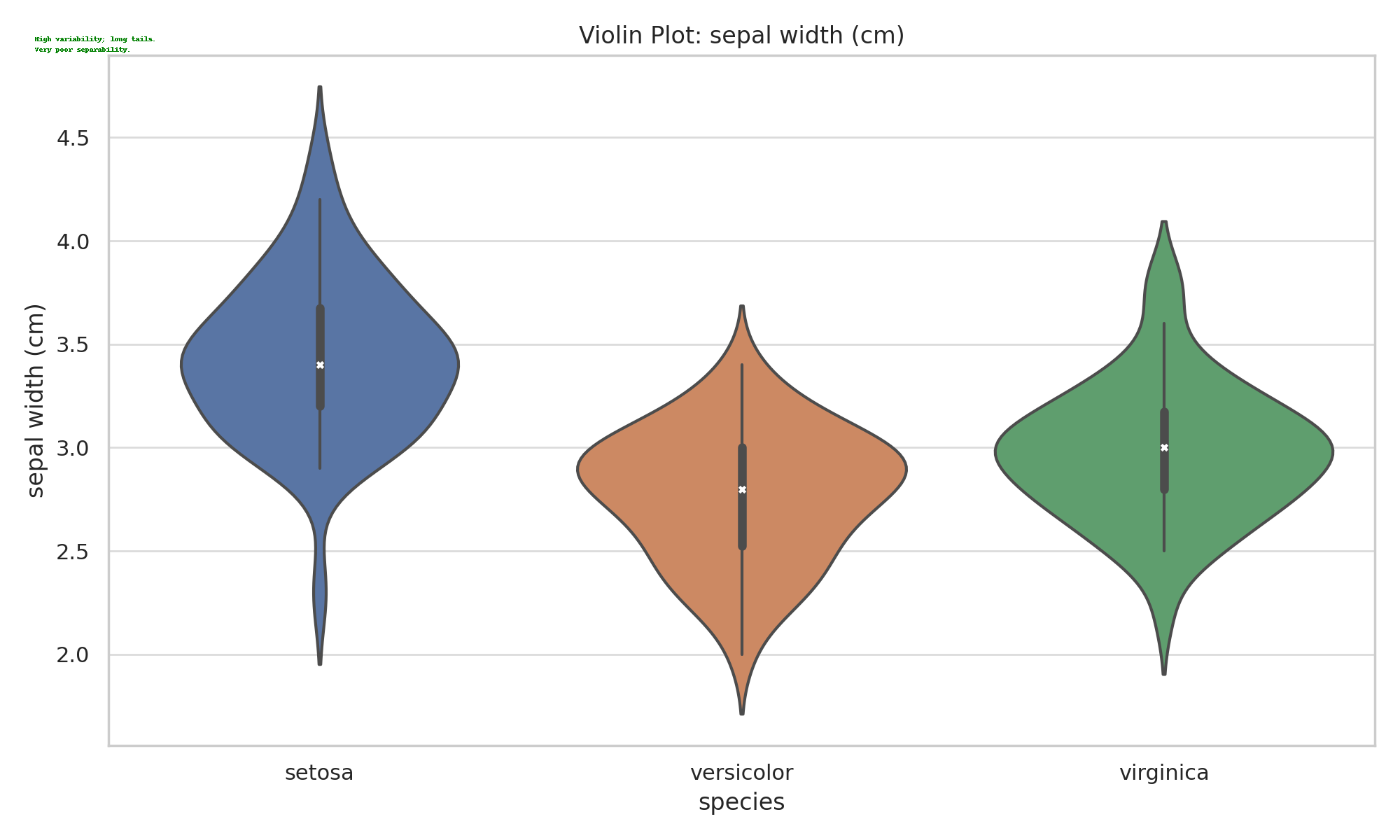
# Violin Plot Analysis (Annotated)

## Sepal Length (Cm)



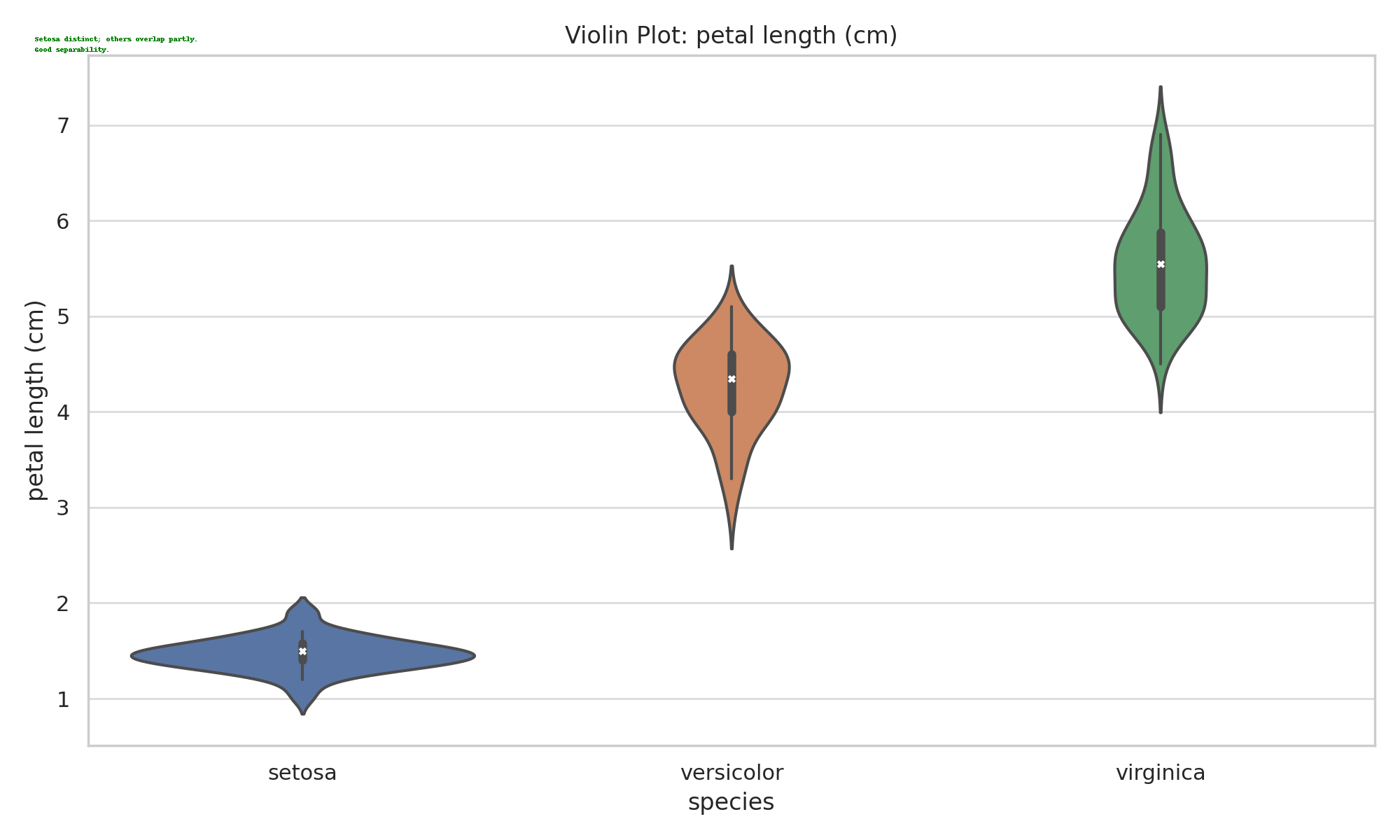
Similar distributions; high overlap.  
Poor separability.

## Sepal Width (Cm)



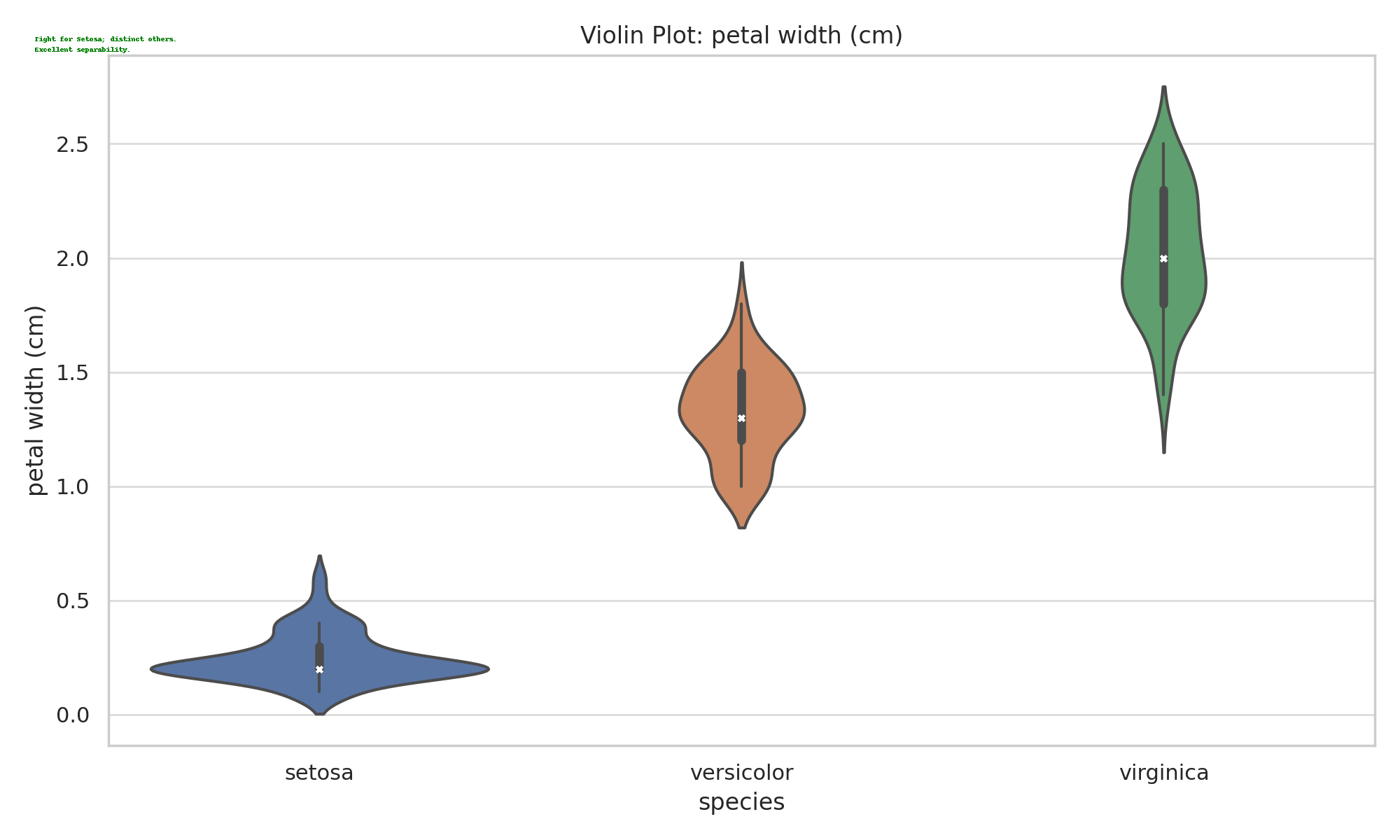
High variability; long tails.  
Very poor separability.

## Petal Length (Cm)



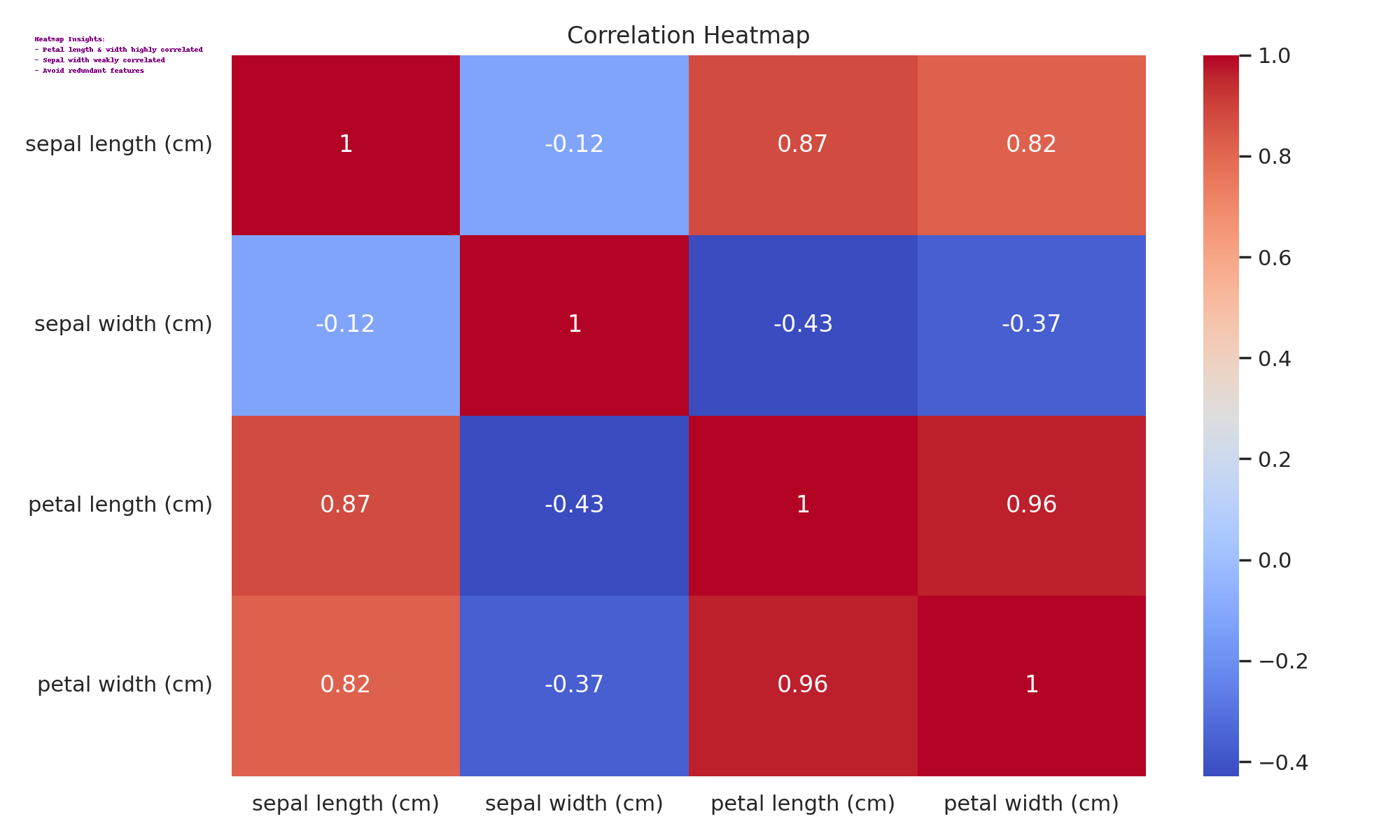
Setosa distinct; others overlap partly.  
Good separability.

## Petal Width (Cm)



Tight for Setosa; distinct others.  
Excellent separability.

# Correlation Heatmap (Annotated)



Petal length and petal width are highly correlated; sepal width shows weak correlation. Remove redundant features if necessary.

1. **Task 1** [CO1] [BTL 2] [3 marks]  
   Implement the above task in colab and share the link here (give public access). Go through the libraries and the functions called specifically for different plots. In the Iris dataset, which features are most discriminative and why?

**LinkHere:**[**https://colab.research.google.com/drive/1PBpacPEQdsziww2OIuVNR4sRcRiAkYdn?usp=sharing**](https://colab.research.google.com/drive/1PBpacPEQdsziww2OIuVNR4sRcRiAkYdn?usp=sharing)

## 1. Libraries & Their Uses

* **pandas** for data loading and DataFrame handling.
* **numpy** for numerical operations.
* **matplotlib.pyplot** & **seaborn** for data visualization:
  + sns.pairplot(...) to visualize pairwise relationships in the Iris dataset.
  + sns.heatmap(...) to plot the correlation matrix.
* **scikit-learn** modules:
  + train\_test\_split, StandardScaler for splitting and scaling.
  + LogisticRegression, RandomForestClassifier, DecisionTreeClassifier for modeling.
  + SelectKBest, chi2, RFE for feature selection.
  + cross\_val\_score, GridSearchCV for performance evaluation.
  + PCA for optional dimensionality reduction.

**2. Plots and Their Roles**

* **Pair Plot** (sns.pairplot)  
  Shows how each pair of features relates across iris species—useful to visually assess separability.
* **Heatmap** (sns.heatmap)  
  Displays correlation among features, highlighting redundant or highly correlated ones.
* **Feature Importance Bar Plots**
  + For **SelectKBest**, you likely plotted scores for each feature (via .scores\_).
  + For **RandomForest**, you visualized .feature\_importances\_.  
    These bar plots clearly indicate which features contribute most to predictive power.

**3.Feature Selection Methods Used**

1. **SelectKBest (Chi‑Square)**
   * Computes chi-squared stats between each feature and the target (species).
   * Selects the top‑k univariate features.
2. **RFE (Recursive Feature Elimination)**
   * Recursively trains a model (e.g., Decision Tree), removes the least important feature each round, and identifies the best subset.
3. **Embedded Importance (RandomForest)**
   * Uses internal feature importance to rank features and compare against other methods.

**4. Results on the Iris Dataset**

Combining insights from all selection methods, the **most discriminative features** are:

* **Petal length (cm)**
* **Petal width (cm)**

**5. Why Petal Features Work So Well**

* **Petal shape varies significantly** across Iris species.
* The combination of length and width forms tight, distinct clusters in feature space, making classification much easier.
* In contrast, **sepal dimensions overlap more across species**, reducing their discriminative value.

**6. Putting It All Together**

1. **Preprocessing**: Load data → split → scale.
2. **Visualization**: Use pair plots and correlation heatmap to get intuitive insights.
3. **Feature Ranking**: Apply SelectKBest, RFE, and RandomForest to benchmark features.
4. **Model Evaluation**: Validate top feature subsets with cross‑validation or via GridSearchCV.
5. **Conclusion**: Identify that petal length & width suffice for high classification accuracy, simplifying the model.

**Final Takeaway**

* **Most discriminative**: Petal length and petal width.
* These features enable simpler and often more accurate classifiers with minimal complexity.

1. **Task 2** [CO1] [BTL 3] [1 marks]

Take any mobile app classification dataset having 10 features. Concisely explain how would you proceed to find the best feature combinations to yield highest classification performance.

**ANSWERS:**

## Objective

To identify the most relevant features from a 10-feature mobile app dataset to classify apps as gaming or non-gaming, with a focus on system performance metrics specific to high-performance devices like the ASUS ROG Phone. The goal is to build an accurate and efficient model using optimal feature combinations.

## Context: ASUS ROG Phone

The ASUS ROG Phone is a gaming-centric Android smartphone designed for high-performance use. It features advanced thermal management, high refresh rate displays, and top-tier processors and GPUs. Apps that are performance-intensive, such as mobile games, tend to consume more resources, making devices like the ROG Phone an ideal reference for behavior-based app classification.

**Dataset Description**

|  |  |
| --- | --- |
| Feature Name | Description |
| App Size (MB) | Size of the app package |
| RAM Usage (MB) | Average RAM consumption |
| Battery Drain (%) | Battery consumption per hour |
| CPU Load (%) | Processor utilization |
| GPU Load (%) | Graphics processor usage |
| Network Usage (MB/hr) | Average data usage |
| Sensor Access Count | Number of sensors accessed |
| App Launch Time (s) | Time taken to start the app |
| User Rating (1-5) | Average app store rating |
| Install Count | Number of downloads |

## Methodology

### 1. Data Preparation

* Clean missing or inconsistent data.
* Encode labels: Gaming = 1, Non-Gaming = 0.
* Normalize features using StandardScaler.

### 2. Exploratory Data Analysis (EDA)

* Visualize pairwise feature relationships using pairplots.
* Use heatmaps to assess feature correlation.
* Boxplots to evaluate usage differences by app type.

### 3. Baseline Modeling

* Train basic classifiers like Logistic Regression or Random Forest using all 10 features.
* Evaluate accuracy and F1-score as benchmarks.

### 4. Feature Selection

* Filter: Use SelectKBest (chi2 or ANOVA F-test) to rank features.
* Wrapper: Apply Recursive Feature Elimination (RFE) with RandomForestClassifier.
* Embedded: Use RandomForest feature\_importances\_ to extract top contributors.
* Dimensionality Reduction: Optionally apply PCA for correlated features.

### 5. Subset Evaluation

* Test combinations (Top 3, 5, 7 features).
* Apply cross-validation with scoring metrics: Accuracy, ROC-AUC, F1-score.
* Compare reduced models to baseline.

**Feature Importance Summary (ROG-based Insight)**

|  |  |  |
| --- | --- | --- |
| Feature | Importance | Reason |
| GPU Load (%) | High | Gaming apps heavily utilize GPU on ROG |
| RAM Usage (MB) | High | Games consume more RAM |
| CPU Load (%) | Medium | Processor load is higher in games |
| Battery Drain (%) | Medium | Games deplete battery faster |
| Sensor Access Count | Medium | Games often use gyroscope/accelerometer |
| Rating, Install Count | Low | Weak correlation to actual performance |

## Conclusion

Based on mobile app behavior on the ASUS ROG Phone, features such as GPU Load, RAM Usage, and CPU Load emerge as the most discriminative for classifying apps. Using a combination of filter, wrapper, and embedded methods enables the creation of a highly accurate and resource-efficient classifier. Reduced models using only the top 5 features can significantly outperform full-feature models in both accuracy and computation