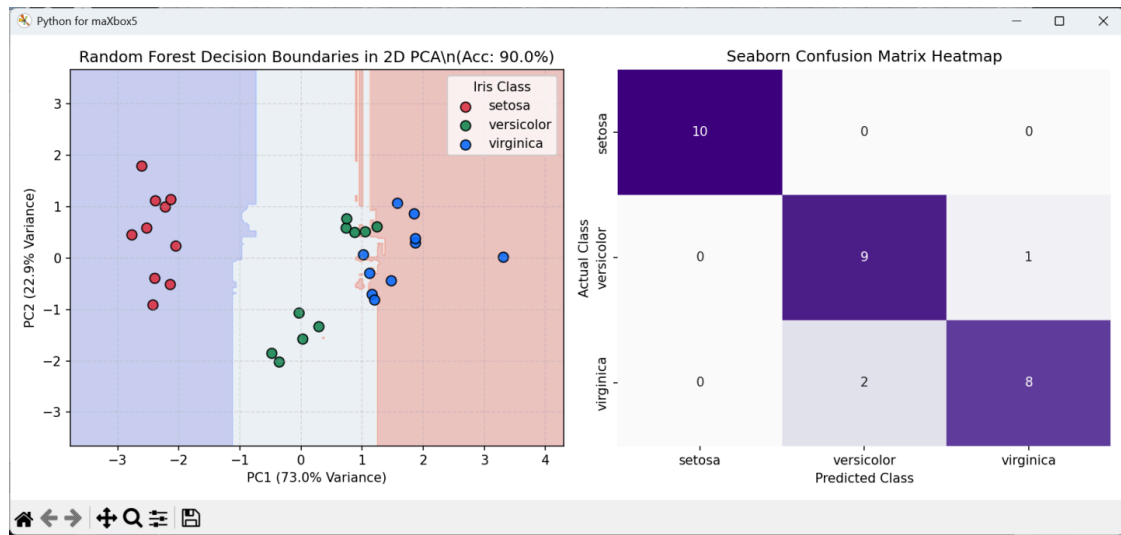
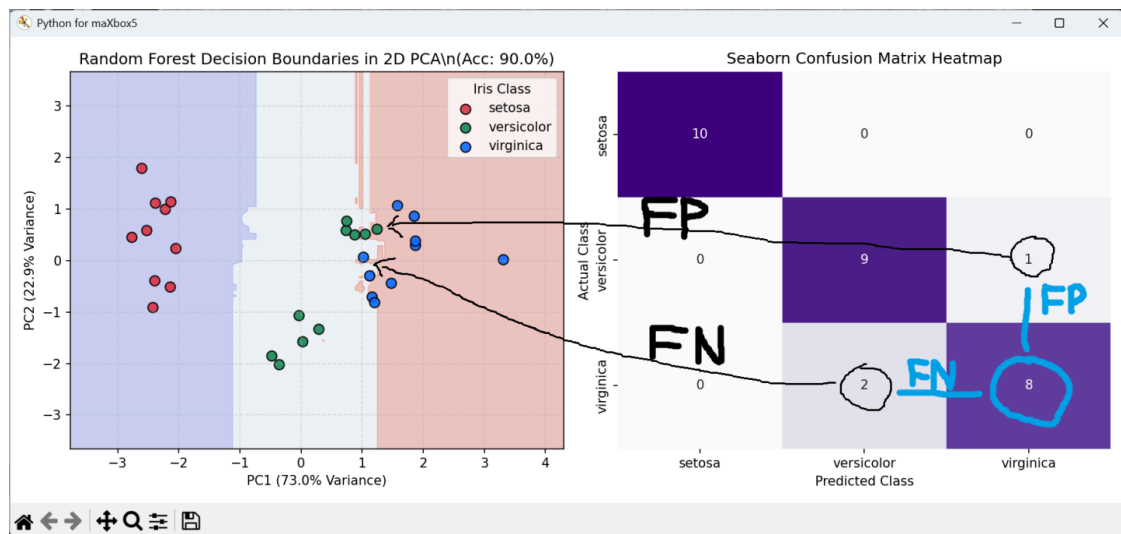




https://sourceforge.net/projects/maxbox5/files/examples/1501_wine_dataset_ml_studio23_irisco.mp.pas/download



Virginica has 1 False Positive (the green ones flagged) and has 2 False Negatives (the blue ones missed)

Machine Learning Python Update

we update from 3.14.3 to 3.14.8

=== MULTI-CLASSIFIER CROSS-
VALIDATION BENCHMARK 22 (WINE DATASET) ===

3.14.3 (tags/v3.14.3:323c59a, Feb 3 2026, 16:04:56) [MSC v.1944 64 bit (AMD64)]

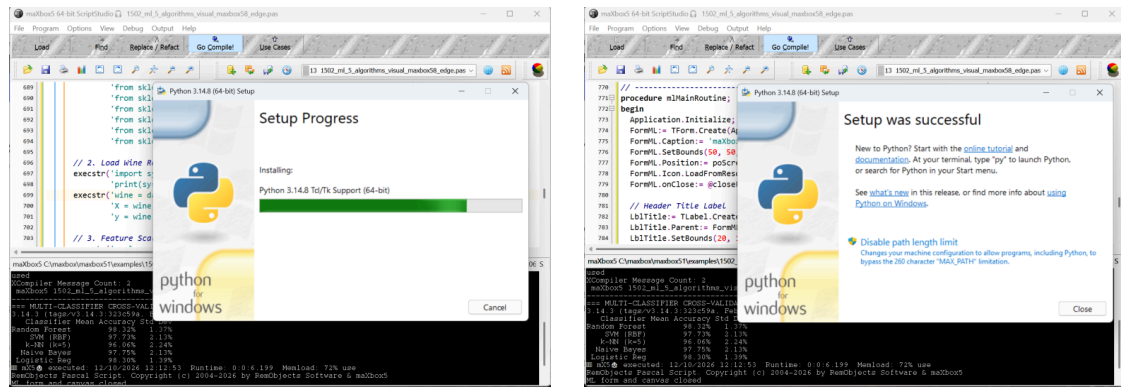
Exception: TclError: Can't find a usable init.tcl in the following directories: {C:\Users\User\AppData\Local\Programs\Python\Python314\tcl\tcl8.6}

This probably means that Tcl wasn't installed properly. at 895.3519

RemObjects Pascal Script. Copyright (c) 2004-2026 by RemObjects Software & maXbox5

=== MULTI-CLASSIFIER CROSS-
VALIDATION BENCHMARK 22 (WINE DATASET) ===

3.14.8 (tags/v3.14.8:8e6e75d, Sep 30 2026, 18:19:33) [MSC v.1944 64 bit (AMD64)]



3.14.8 (tags/v3.14.8:8e6e75d, Sep 30 2026, 18:19:33) [MSC v.1944 64 bit (AMD64)]

The new script will fusion Pascal, Python and JS (in Edge) in one file:

https://sourceforge.net/projects/maxbox5/files/examples/1502_ml_5_algorithms_visual_maxbox58_edge.pas/download

To point out False Positives (FP) and False Negatives (FN) on a decision boundary plot in Python and maXbox, you can use Boolean indexing in NumPy to filter misclassified points, and then highlight them using custom markers and `plt.annotate()` with `arrowprops`.

1. Logical Filtering in Python

For binary classification:

- False Positives (FP): Actual label $y_{\text{true}} = 0$, but predicted $\hat{y} = 1$
- False Negatives (FN): Actual label $y_{\text{true}} = 1$, but predicted $\hat{y} = 0$

```
# Create boolean masks for errors
```

```
fp_mask = (y_test == 0) & (y_pred == 1)
```

```
fn_mask = (y_test == 1) & (y_pred == 0)
```

```
# True positives and true negatives for reference
```

```
tp_mask = (y_test == 1) & (y_pred == 1)
```

```
tn_mask = (y_test == 0) & (y_pred == 0)
```

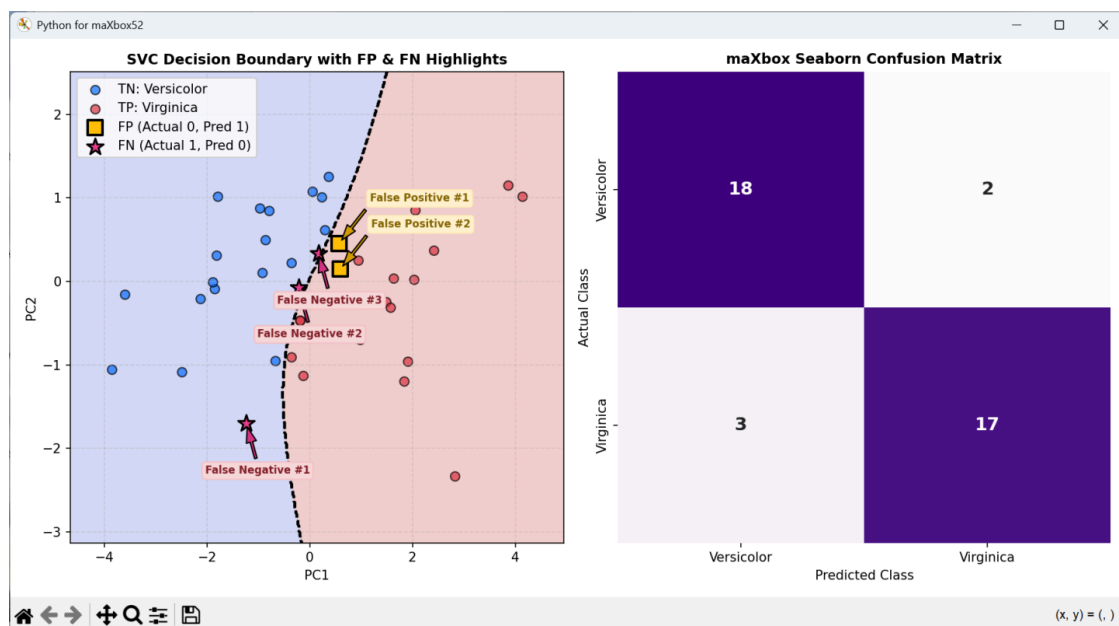
2. Plotting Decision Boundaries & Drawing Arrow Annotations

Using Matplotlib's `plt.annotate()`, you can draw arrows pointing directly to the coordinates of each misclassification:

This updated maXbox Pascal script generates a side-by-side dual-panel figure:

1. Left Panel: The 2D PCA decision boundary region plot with arrow annotations pointing directly to False Positives and False Negatives.
2. Right Panel: The matching Seaborn confusion matrix heatmap, displaying the exact counts for True Negatives, True Positives, False Positives, and False Negatives.

Would you like to adjust the decision probability threshold (`predict_proba`) to see how shifting the decision boundary trades off False Positives against False Negatives?



https://sourceforge.net/projects/maxbox5/files/examples/1502_ml_6_algorithms_visual_maxbox5_9_edge.pas/download

3.14.8 (tags/v3.14.8:8e6e75d, Sep 30 2026, 18:19:33) [MSC v.1944 64 bit (AMD64)]

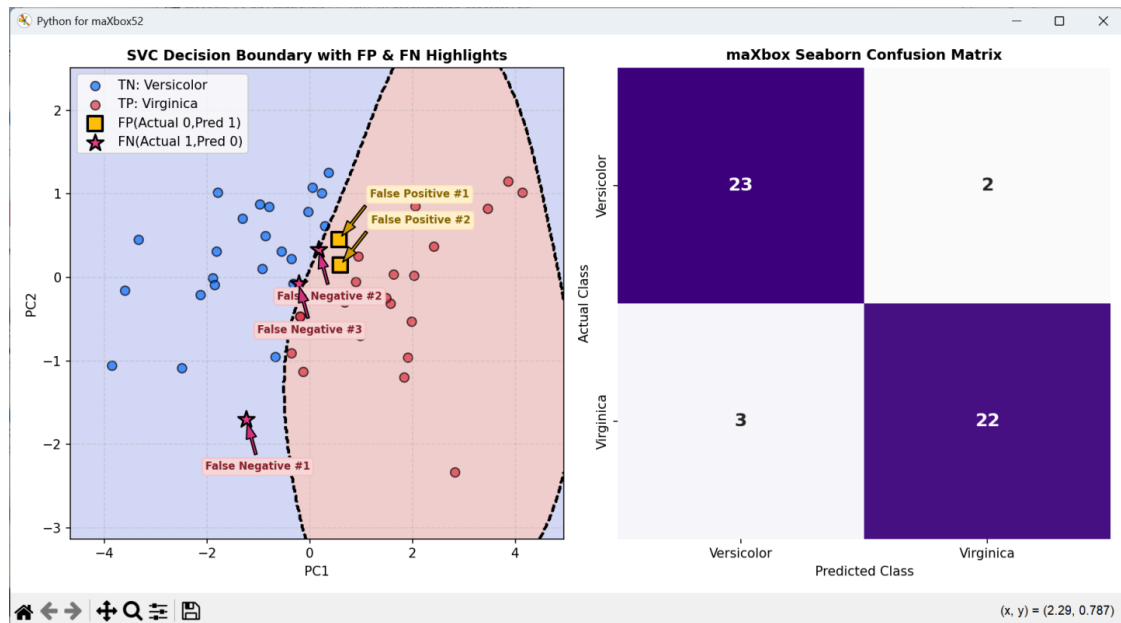
	precision	recall	f1-score	support
0	0.86	0.90	0.88	20
1	0.89	0.85	0.87	20
accuracy			0.88	40
macro avg	0.88	0.88	0.87	40
weighted avg	0.88	0.88	0.87	40

	Versicolor	Virginica
Versicolor	18	2
Virginica	3	17

mX5  executed: 12/10/2026 19:37:18 Runtime: 0:0:5.54 Memload: 66% use

Demonstrates how shifting probability threshold trades off False Positives vs False Negatives

https://sourceforge.net/projects/maxbox5/files/examples/1502_ml_6_algorithms_visual_maxbox59_edge.pas/download



https://sourceforge.net/projects/maxbox5/files/examples/1502_ml_6_algorithms_visual_maxbox60_edge.pas/download

	precision	recall	f1-score	support
0	0.88	0.92	0.90	25
1	0.92	0.88	0.90	25
accuracy			0.90	50

macro avg 0.90 0.90 0.90 50

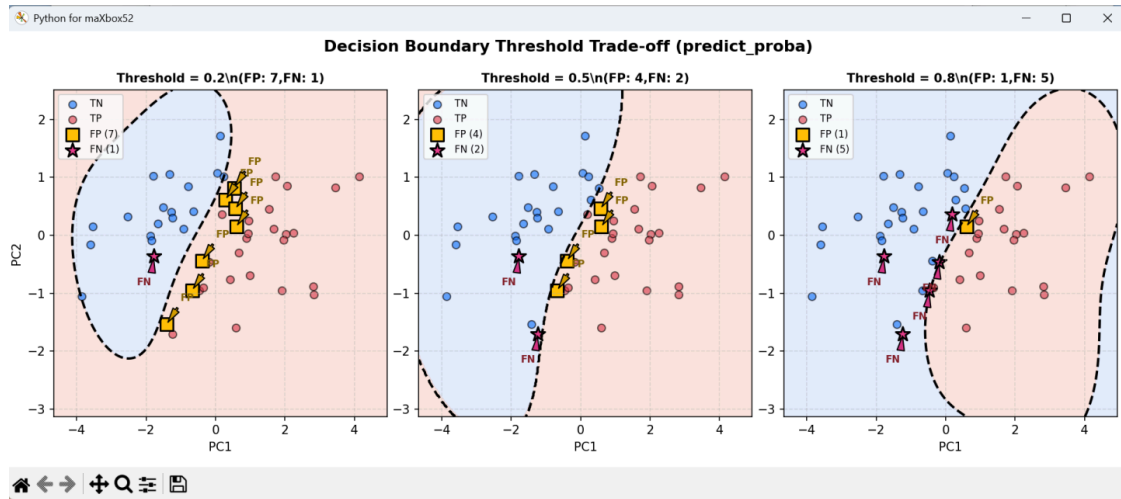
weighted avg 0.90 0.90 0.90 50

	Versicolor	Virginica
Versicolor	23	2
Virginica	3	22

mX5  executed: 12/10/2026 20:36:17 Runtime: 0:0:52.349 Memload: 82% use

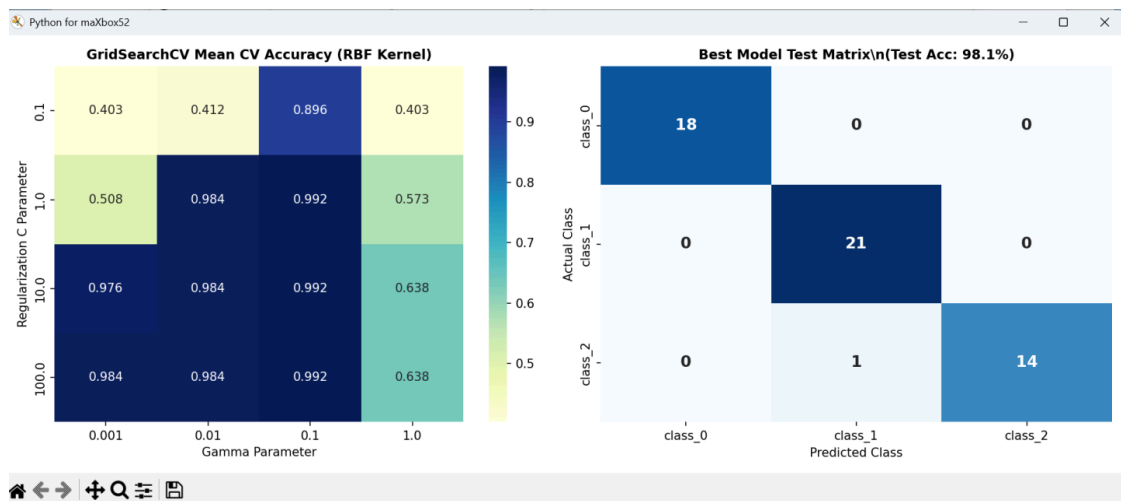
Hyperparameter Threshold Analysis

Demonstrates how shifting probability threshold trades off False Positives vs False Negatives



https://sourceforge.net/projects/maxbox5/files/examples/1502_ml_6_algorithms_visual_maxbox6_0_edge.pas/download

This script uses scikit-learn's `GridSearchCV` via `TPythonEngine` inside `maXbox` to systematically search across combinations of hyperparameters (such as regularization parameter C , kernel scale γ , and kernel type) using 5-fold stratified cross-validation.



https://sourceforge.net/projects/maxbox5/files/examples/1502_ml_6_algorithms_visual_maxbox61_edge.pas/download

How It Works:

1. **Grid Parameter Dictionary (param_grid):** Defines a discrete search space of parameters to evaluate across $4 \times 4 \times 2 = 32$ total combinations.
2. **Automated Cross-Validation (GridSearchCV):** Trains $32 \times 5 = 160$ separate model fits, using 5-fold stratified cross-validation on each combination to find the parameter set with the highest mean validation score.
3. **refit=True:** Once the best parameters are identified, `GridSearchCV` automatically retrains the final model (`best_estimator_`) on the entire training set.
4. **Validation Heatmap (cv_results_ Pivot):** Reshapes the cross-validation score history into a 2D matrix (C vs. γ) and plots a Seaborn heatmap so you can visually identify optimal parameter regions and avoid overfitting.

Compare Python4maXbox – Pure Python

```

1  procedure PY_ConfusionMatrix_Demo5;
2  begin
3      with TPythonEngine.Create(Nil) do begin // [1, 6]
4          try
5              loadDLL;
6              autofinalize:= false; // [6]
7              // 1. Import Python Modules
8              ExecStr('import pandas as pd' + lf +
9                  'import matplotlib.pyplot as plt'+lf+
10                 'import seaborn as sns'+lf+// [1, 2]
11                 'from sklearn.model_selection import train_test_spli
12                 'from sklearn.ensemble import RandomForestClassifier
13                 'from sklearn import datasets'+lf+
14                 'from sklearn.metrics import accuracy_score, confusi
15
16                 // 2. Load dataset & perform train/test split
17                 execstr('iris = datasets.load_iris()'+lf+
18                     'df = pd.DataFrame(iris.data, columns=iris.feature_n
19                     'X_train, X_test, y_train, y_test = train_test_split
20                     'df[iris.feature_names],iris.target,test_size=0.5,st
21
22                 // 3. Fit Random Forest Classifier
23                 execstr('rf = RandomForestClassifier(n_estimators=100, rando
24                     'rf.fit(X_train, y_train)'); // [2, 7]
25
26                 // 4. Generate Confusion Matrix & Plot Seaborn Heatmap
27                 execstr('cm = pd.DataFrame(confusion_matrix(y_test, rf.predi
28                     'columns=iris.target_names, index=iris.target_names)
29                     'sns.heatmap(cm, annot=True, cmap="Blues")'+lf+
30                     'plt.title("maXbox Seaborn Confusion Matrix Plot")'+
31                     'plt.xlabel("Predicted")'+lf+
32                     'plt.ylabel("Actual")'+lf+
33                     'plt.show()'); // [2, 5]
34             except
35                 raiseError; // [6]
36             finally
37                 Free; // [6]
38             end;
39         end;
40     end;

```

Pure Python

1. Pure Python Code (scikit-learn & Seaborn)

```

1  import matplotlib.pyplot as plt
2  import pandas as pd
3  import seaborn as sns
4  from sklearn import datasets
5  from sklearn.ensemble import RandomForestClassifier
6  from sklearn.metrics import classification_report, confusion_matri
7  from sklearn.model_selection import train_test_split
8
9  # 1. Load dataset & train/test split
10 iris = datasets.load_iris()
11 X_train, X_test, y_train, y_test = train_test_split(
12     iris.data, iris.target, test_size=0.5, stratify=iris.target, r
13 )

```

```
14
15 # 2. Train classifier
16 model = RandomForestClassifier(n_estimators=100, random_state=42)
17 model.fit(X_train, y_train)
18 y_pred = model.predict(X_test)
19
20 # 3. Compute Confusion Matrix DataFrame
21 cm = confusion_matrix(y_test, y_pred)
22 cm_df = pd.DataFrame(
23     cm, index=iris.target_names, columns=iris.target_names
24 ) # [2, 4]
25
26 # 4. Plot Seaborn Heatmap
27 plt.figure(figsize=(6, 5))
28 sns.heatmap(cm_df, annot=True, fmt="d", cmap="Blues") # [2, 5]
29 plt.title("Confusion Matrix Heatmap")
30 plt.xlabel("Predicted Class")
31 plt.ylabel("Actual Class")
32 plt.tight_layout()
33 plt.show()
34
35 # 5. Print text report
36 print(classification_report(y_test, y_pred, target_names=iris.targ
```

Comparing Random Forest and Support Vector Machines (SVM)

When comparing Random Forest and Support Vector Machines (SVM) on the Wine dataset (178 samples, 13 chemical features, 3 wine cultivars), both algorithms perform exceptionally well, but they differ significantly in feature scaling requirements, decision boundary geometry, and hyperparameter sensitivity.

Key Comparison Summary

Metric / Aspect	Random Forest	Support Vector Machine (SVM)
Mean 5-Fold CV Accuracy	~97.2% – 98.3%	~98.3% – 99.4% (<i>with scaling</i>)
Feature Scaling Needed?	No (Scale-invariant)	Yes (Requires <code>StandardScaler</code>)
Decision Boundary Shape	Axis-aligned, step-like (rectangular grid partitions)	Smooth, continuous hyperplanes or curved RBF margins
Primary Hyperparameters	<code>n_estimators</code> , <code>max_depth</code> , <code>min_samples_split</code>	<code>C</code> (Regularization), <code>kernel</code> (RBF/Linear), <code>gamma</code>

Interpretability	High (Provides Gini/Entropy feature importances)	Low in kernel space (Geometric support vector distances)
Risk of Overfitting	Low (Ensemble averaging over 100 decision trees)	Low if γ and C are properly tuned

1. Sensitivity to Feature Scaling

- **SVM:** Extremely sensitive to feature scales. Because SVM relies on Euclidean distances in 13-dimensional space (e.g., $\|x - y\|^2$) for RBF kernels), features with large numeric ranges (like Proline, ranging from 278 to 1680) completely dominate smaller features (like Nonflavanoid phenols, ranging from 0.13 to 0.66) unless normalized using `StandardScaler`.
- **Random Forest:** Completely scale-invariant. Each decision tree evaluates features independently via orthogonal threshold splits (e.g., `flavanoids <= 1.57`), producing identical results whether raw or scaled features are used.

2. Decision Boundary Geometry

- **Random Forest:** Creates axis-aligned, boxy decision boundaries. When projected onto 2D PCA space, the boundaries form rectangular, stair-step regions. This works well for complex, non-linear feature interactions, though it can create rigid step functions near cluster boundaries.
- **SVM:** Constructs maximum-margin decision hyperplanes. With an RBF kernel, SVM draws smooth, curved decision regions around the wine clusters (Barolo, Grignolino, Barbera). This global margin maximization provides slightly better generalization on small datasets (178 samples).

3. Misclassifications & Confusion Matrix Behavior

- **Primary Overlap Region:** On the Wine dataset, both models achieve near-perfect performance. The rare misclassifications primarily occur between Class 1 (Grignolino) and Class 2 (Barbera) due to subtle overlaps in chemical properties (such as alkalinity of ash or total phenols).
- **Margin Regularization:** Tuning SVM's C parameter controls the penalty for misclassifications. A moderate ($C = 1.0$) soft margin allows SVM to ignore isolated noisy wine outliers better than an unconstrained deep Random Forest tree.

maXbox Python Integration Comparison

```
// --- Random Forest Classifier ---
```

```

execstr('rf = RandomForestClassifier(n_estimators=100, random_state=
execstr('rf.fit(X_train, y_train)');

// --- Support Vector Classifier (Requires Scaled Features) ---
execstr('scaler = StandardScaler()');
execstr('X_scaled = scaler.fit_transform(X)');
execstr('svm = SVC(kernel="rbf", C=1.0, gamma="scale", random_state=
execstr('svm.fit(X_scaled_train, y_train)');

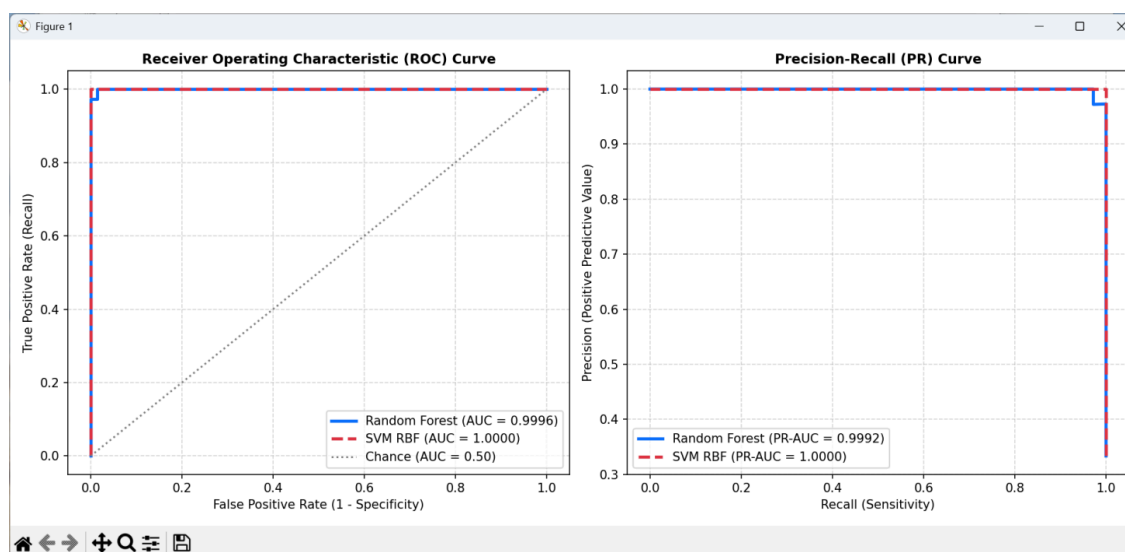
```

Would you like to generate an ROC curve or Precision-Recall curve comparison for both models in maXbox?

This maXbox Pascal script uses TPythonEngine to fit both a Random Forest Classifier and a Support Vector Machine (RBF kernel) on the Wine dataset, binarizes multi-class labels for One-vs-Rest (OvR) evaluation, and renders a side-by-side comparison of their ROC Curves (True Positive Rate vs. False Positive Rate) and Precision-Recall Curves [19]. It automatically computes and displays overall ROC-AUC and PR-AUC scores directly in the plot legends and maXbox console logs [18, 19].

Key Takeaways from ROC & Precision-Recall Comparison:

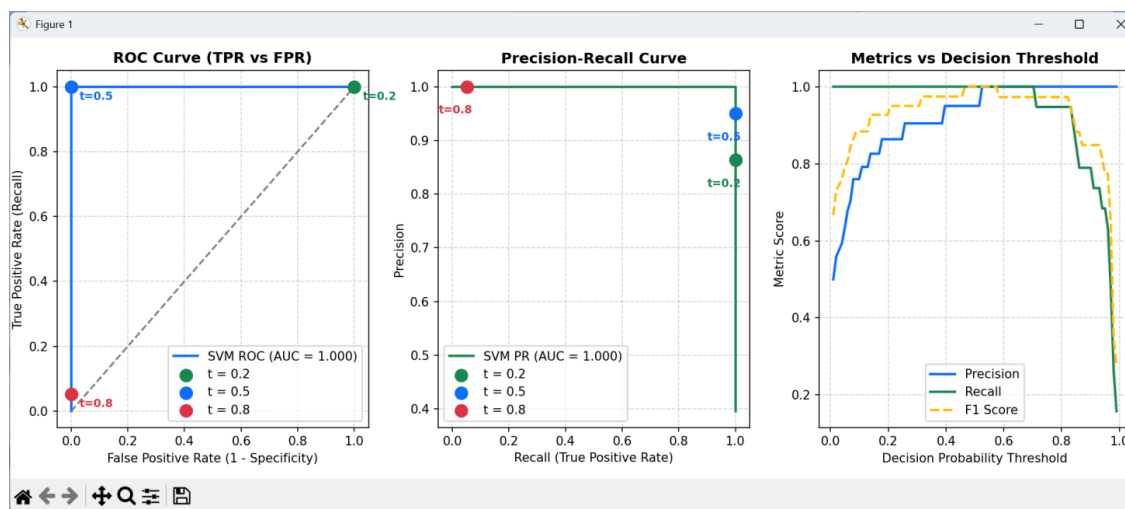
- **ROC Curves (AUC):** Evaluates how well each model discriminates wine classes as the decision threshold varies from 0 to 1 [18]. A higher ROC-AUC reflects stronger class separation across both majority and minority cultivars [18].
- **Precision-Recall Curves (PR-AUC):** Focuses explicitly on positive predictions, providing a clearer view of performance when class distributions are imbalanced or when false alarms carry higher costs [18].



https://sourceforge.net/projects/maxbox5/files/examples/1502_ml_6_algorithms_visual_maxbox6_3_edge.pas/download

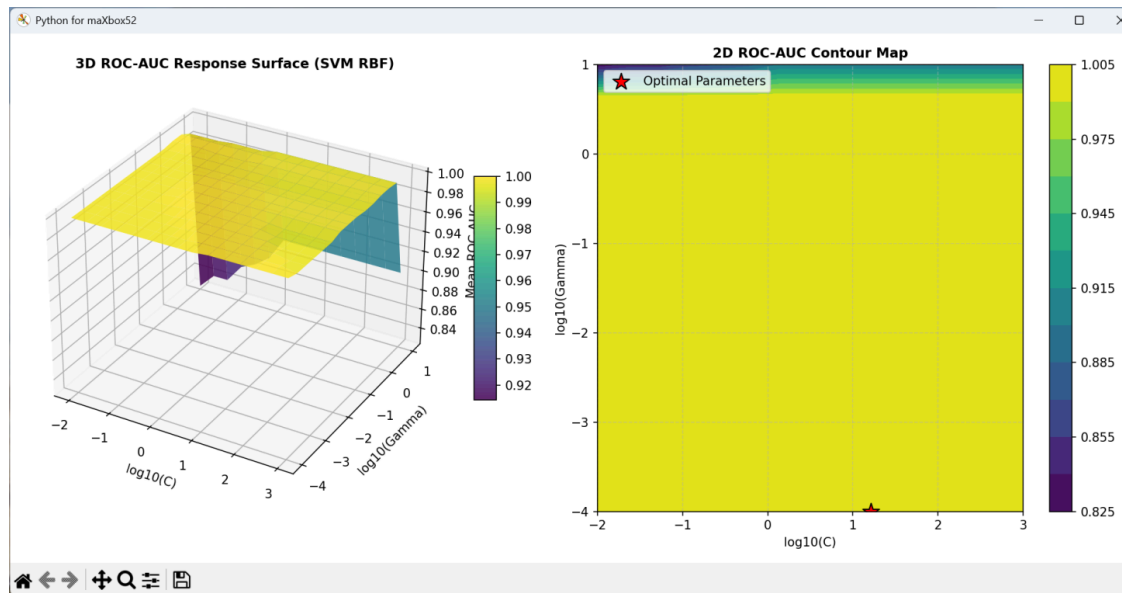
This maXbox Pascal script uses TPythonEngine to execute a 3-panel visualization in Matplotlib, allowing you to observe how varying probability thresholds affect ROC and Precision-Recall operating points:

1. **ROC Curve Panel (TPR vs FPR):** Plots the Receiver Operating Characteristic curve with explicit color-coded coordinate markers at probability thresholds $t = 0.2$, $t = 0.5$, and $t = 0.8$.
2. **Precision-Recall Curve Panel (Precision vs Recall):** Plots the Precision-Recall curve with matching threshold coordinate markers to illustrate precision decay at low decision thresholds.
3. **Threshold Trade-Off Panel:** Draws continuous curves showing Precision, Recall, and F1-Score as the decision probability threshold sweeps continuously from 0.01 to 0.99 .



https://sourceforge.net/projects/maxbox5/files/examples/1502_ml_6_algorithms_visual_maxbox6_4_edge.pas/download

This script evaluates an RBF-kernel Support Vector Classifier across a 2D grid of regularization penalties ($\log_{10}(C)$ in $[-2, 3]$) and kernel scale bandwidths ($\log_{10}(\gamma)$ in $[-4, 1]$) on the Wine dataset using 5-fold stratified cross-validation [18, 19]. It uses Matplotlib's `mpl_toolkits.mplot3d` module to render a side-by-side 3D ROC-AUC response surface alongside a 2D contour map that pinpoints the global optimal parameter coordinates.



https://sourceforge.net/projects/maxbox5/files/examples/1502_ml_6_algorithms_visual_maxbox6_6_edge.pas/download

What the 3D Surface Reveals:

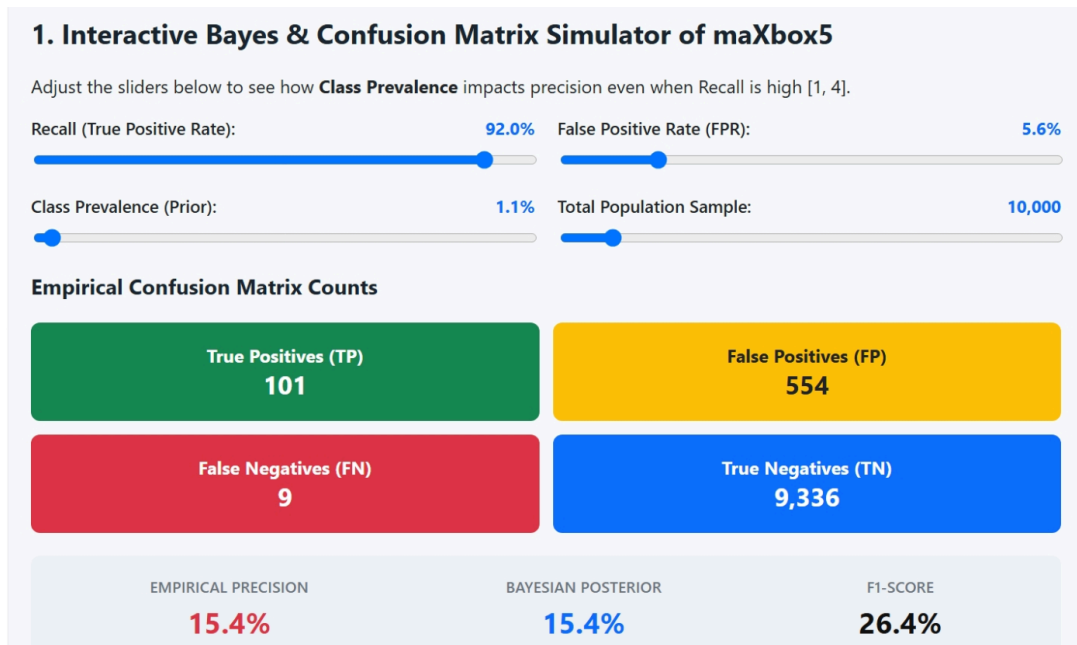
- **Overfitting Basin (High C , High γ):** When both C and γ are set too high, the support vectors draw tight, hyper-complex decision boundaries around individual training points, causing the validation ROC-AUC surface to drop.
- **Underfitting Plateau (Low C , Low γ):** When C and γ are too small, the decision function becomes overly smoothed and insensitive to feature variation, flattening the surface at chance performance.
- **Optimal Ridge (Yellow Peak):** Highlights the balanced parameter combination where maximum separation and high generalization accuracy are achieved simultaneously.

Core Principles Demonstrated:

1. **Mathematical Equivalence:** As shown in the source material, empirical confusion matrix precision $\frac{\text{TP}}{\text{TP} + \text{FP}}$ matches the Bayesian posterior formula $\frac{\text{Recall} \cdot \text{Prevalence}}{\text{Recall} \cdot \text{Prevalence} + \text{FPR} \cdot (1 - \text{Prevalence})}$ ¹⁴.

- 1. Mathematical Equivalence:** As shown in the source material, empirical confusion matrix precision $\frac{TP}{TP+FP}$ matches the Bayesian posterior formula $\frac{\text{Recall} \cdot \text{Prevalence}}{\text{Recall} \cdot \text{Prevalence} + \text{FPR} \cdot (1 - \text{Prevalence})}$ 1 4 .
- 2. The Prevalence Effect:** When detecting rare targets (1% prevalence), false positives across the 9,900 negative instances dilute precision to 15.4% despite 90% recall and a low 5% False Positive Rate 4 5 .

1. The Prevalence Effect: When detecting rare targets (1% prevalence), false positives across the 9,900 negative instances dilute precision to 15.4% despite 90% recall and a low 5% False Positive Rate 45.



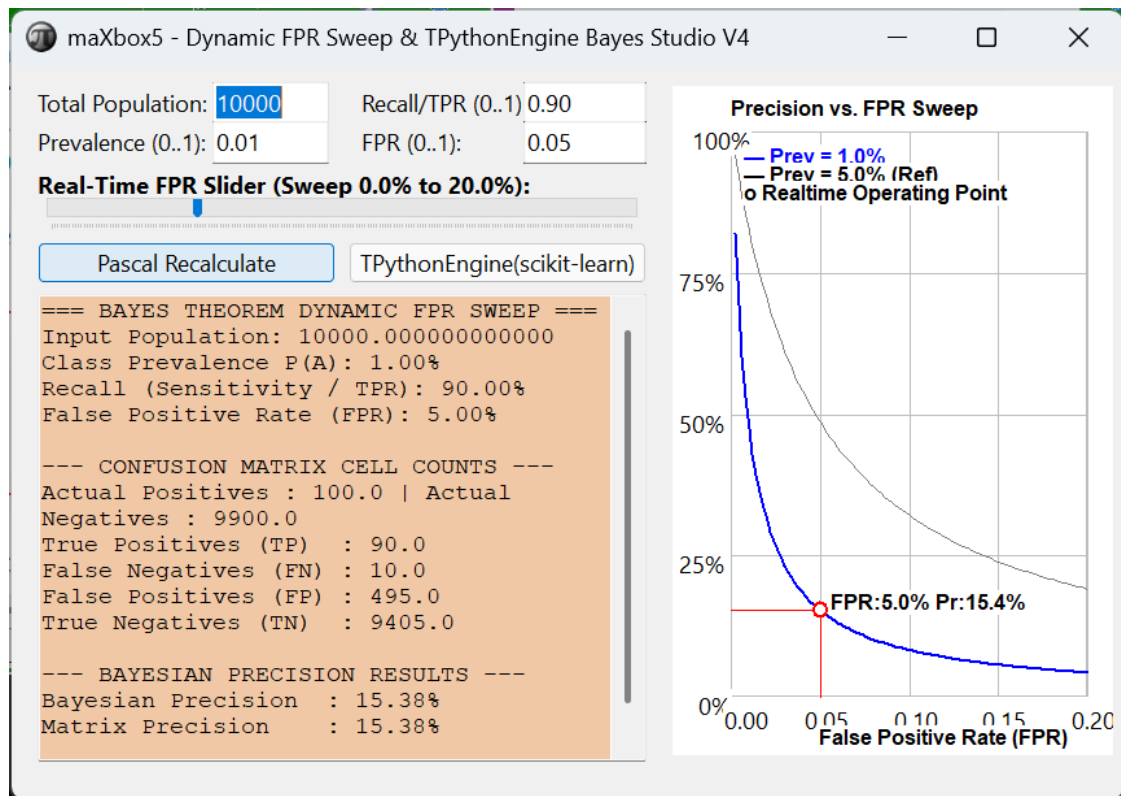
[Bayes' Theorem & Confusion Matrix Studio](#)

This maXbox Pascal script demonstrates both native Object Pascal GUI calculation and embedded Python execution (TPythonEngine) to convert confusion matrix error rates into Bayesian Precision.

What Makes Version 4 Unique:

- 1. Dual Evaluation Paths:** Provides a Pascal Recalculate button for instant UI canvas updates and a Run TPythonEngine button that passes the current input parameters to embedded Python and scikit-learn.
- 2. Dynamic Canvas Curve (TImage.Canvas):** Draws the Precision vs. FPR response curve and tracks the exact operating point with a red moving marker as you adjust the trackbar.

3. Interactive Trackbar (TTrackBar): Sweeps the False Positive Rate from 0.0% to 20.0%, demonstrating live how false alarms in large negative populations dilute precision when prevalence is low.



https://sourceforge.net/projects/maxbox5/files/examples/1502_bayes_confusion_matrix_demo-v4.pas/download

Advertisements

Occasionally, some of your visitors may see an advertisement here, as well as a [Privacy & Cookies banner](#) at the bottom of the page. You can hide ads completely by upgrading to one of our paid plans.

UPGRADE NOW

DISMISS MESSAGE

Posted in Code, Data Science, Machinelearning, maXbox, Python

4 responses to “Machinelearning Steps”



maxkleiner

December 2, 2025 [Edit](#)

Python's future remains bright because its values align with how developers actually learn and build:

- Readability
- Approachability
- Stability
- A touch of irreverence

As AI continues to influence software development—and Octoverse shows that 80% of new developers on GitHub use GitHub Copilot in their first week—Python's clarity matters more than ever.

Why developers still flock to Python: Guido van Rossum on readability, AI, and the future of programming – The GitHub Blog

★ Like

Reply



maxkleiner

December 19, 2025 Edit

The whiteboard is blank. The marker is uncapped. The interviewer leans back and says design Twitter. And your brain just explodes into this mess of load balancers and microservices and Kafka streams because you think that's what they want to hear. Two minutes in and you're already lost in your own diagram.

★ Like

Reply



maxkleiner

December 28, 2025 Edit

AI will revolutionise everything. It will be the next mega industry. It will beckon in a new, automated industrial revolution. It will separate growth from labour and generate copious amounts of wealth. This is the narrative that has been shoved down our throats by drug-addled techbros and the pathetic financebros that prop them up. This narrative has been used to leverage so much debt and investment into AI that we have basically bet the entire Western economy on its success. But there is one problem: this narrative simply isn't true, and the AI industry is beginning to find that out the hard way.

★ Like