# Exercise 2: Performance metrics for Classifiers

## Objective

To implement Logistic Regression, Decision Tree, and SVM classifiers and evaluate them using multiple performance metrics on a complex dataset.

## Introduction

Classifier performance cannot be assessed by accuracy alone, especially with imbalanced datasets. Metrics such as precision, recall, F1-score, ROC-AUC, and PR-AUC provide deeper insights into model performance. This exercise compares three classifiers—Logistic Regression, Decision Tree, and SVM—on a high-dimensional, imbalanced dataset.

## Dataset

We use sklearn's make_classification to generate a dataset with 3000 samples and 50 features, including informative, redundant, and noisy features with class imbalance (3 classes, 60%, 30%, 10%).

## Procedure

1. Generate a high-dimensional imbalanced dataset.
2. Split into training and testing sets.
3. Train Logistic Regression, Decision Tree, and SVM classifiers.
4. Predict class labels and probabilities.
5. Evaluate using Accuracy, Precision, Recall, F1-score, ROC-AUC, and PR-AUC.
6. Plot ROC and PR curves for the minority class.

## Python Program

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler,
label_binarize
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.metrics import (
    accuracy_score, precision_score, recall_score, f1_score,
    roc_auc_score, average_precision_score, roc_curve,
precision_recall_curve
)

# Dataset
X, y = make_classification(
    n_samples=3000, n_features=50, n_informative=15,
n_redundant=10,
    n_classes=3, weights=[0.6, 0.3, 0.1], flip_y=0.02,
random_state=42
)
X_train, X_test, y_train, y_test = train_test_split(X, y,
stratify=y, test_size=0.2, random_state=0)

# Classifiers
classifiers = {
    "Logistic Regression": LogisticRegression(max_iter=2000,
multi_class='multinomial'),
    "Decision Tree": DecisionTreeClassifier(max_depth=10,
random_state=42),
    "SVM": SVC(kernel='rbf', probability=True)
}

# Evaluation
for name, clf in classifiers.items():
    if name in ["Logistic Regression", "SVM"]:  # scale
features
        scaler = StandardScaler()
        X_train_scaled = scaler.fit_transform(X_train)
        X_test_scaled = scaler.transform(X_test)
        clf.fit(X_train_scaled, y_train)
        y_pred = clf.predict(X_test_scaled)
        y_proba = clf.predict_proba(X_test_scaled)
    else:
        clf.fit(X_train, y_train)
        y_pred = clf.predict(X_test)
        y_proba = clf.predict_proba(X_test)

    acc = accuracy_score(y_test, y_pred)
    prec = precision_score(y_test, y_pred, average='macro')
    rec = recall_score(y_test, y_pred, average='macro')
    f1 = f1_score(y_test, y_pred, average='macro')

    y_bin = label_binarize(y_test, classes=[0,1,2])
    roc_auc = roc_auc_score(y_bin, y_proba, average='macro',
multi_class='ovr')
    pr_auc = average_precision_score(y_bin, y_proba,
average='macro')

    print(f"\n{name}")
    print(f"Accuracy: {acc:.3f}")
    print(f"Precision (macro): {prec:.3f}")
    print(f"Recall (macro): {rec:.3f}")
    print(f"F1-score (macro): {f1:.3f}")
    print(f"ROC-AUC (macro): {roc_auc:.3f}")
    print(f"PR-AUC (macro): {pr_auc:.3f}")

# ROC & PR for minority class (class 2) with SVM
y_bin = label_binarize(y_test, classes=[0,1,2])
clf = SVC(kernel='rbf', probability=True)
X_train_scaled = StandardScaler().fit_transform(X_train)
X_test_scaled = StandardScaler().fit_transform(X_test)
clf.fit(X_train_scaled, y_train)
y_proba = clf.predict_proba(X_test_scaled)
fpr, tpr, _ = roc_curve(y_bin[:,2], y_proba[:,2])
prec, rec, _ = precision_recall_curve(y_bin[:,2],
y_proba[:,2])

plt.figure(figsize=(12,5))
plt.subplot(1,2,1)
plt.plot(fpr, tpr, label='ROC Curve (class 2)')
plt.plot([0,1],[0,1],'--')
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('ROC Curve for Minority Class')
plt.legend()
plt.subplot(1,2,2)
plt.plot(rec, prec, label='PR Curve (class 2)')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall Curve for Minority Class')
plt.legend()
plt.tight_layout()
plt.show()
```

## Tasks

Task 1 [CO1] [BTL 3] [2 marks]
Run the code and report all metrics for the three classifiers. Paste outputs. Change dataset imbalance (e.g., weights=[0.7,0.2,0.1]) and observe metric differences across classifiers.

**Output : 2025007855 - Task 1**

### Inference for Task-1 :

When we change dataset imbalance to weights=[0.7,0.2,0.1] . We can notice that accuracy for all classifiers show a slight increase in accuracy with the more imbalanced dataset.

On the other side, we can observe that metrics like macro-precision, recall and F1 decreased across the board after increasing imbalance .

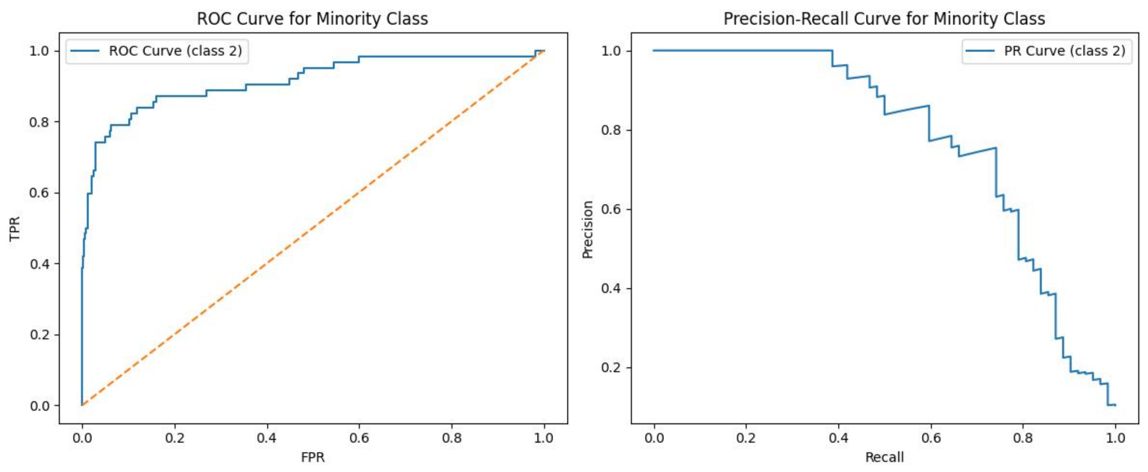
Table : Metrics after changing dataset imbalance

| Models/Metrics | Logistic Regression | Decision Tree | SVM |
|---|---|---|---|
| **Accuracy** | 79% | 76% | 87% |
| **Precision(macro)** | 70% | 65% | 90% |
| **Recall(macro)** | 59% | 61.1% | 69% |
| **F1-Score(macro)** | 63% | 62.6% | 76.1% |
| **ROC-AUC(macro)** | 82% | 71% | 92.3% |
| **PR-AUC(macro)** | 67% | 52.6% | 85.6% |

While , SVM retains the highest macro scores, showing better balance across classes.

Now, changes in the ROC-AUC and PR-AUC  upon changing weight imbalance , ROC-AUC dropped slightly for all models while PR-AUC shows a more noticable decline.

But, SVM again led in both metrics (i.e., ROC-AUC,PR-AUC) shows the strength of probability seperation .



**Conclusion :** Accuracy alone is misleading under imbalance suggests that macro metrics and PR curves are essential. SVM consistently outperformed when compared to logistic regression and decision tree.

Task 2 [CO1] [BTL 5] [3 marks]
Plot ROC & PR curves for Logistic Regression and Decision Tree. Compare with SVM and discuss which classifier handles imbalance better.

**Output : 2025007855 - Task-2**

**Inference for Task-2 :**

On plotting ROC and PR curves for Logistic regression and decision tree now on comparing these two models with SVM metrics:

**Logistic Regression** demonstrates the strongest performance under the imbalanced setting [0.7, 0.2, 0.1].

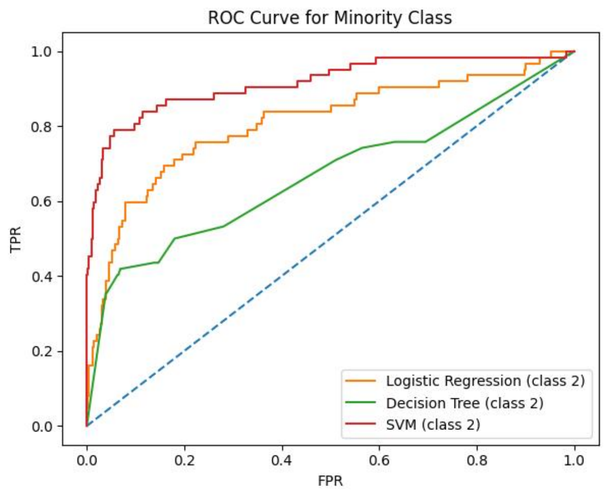
- ✓ Its ROC curve is closest to the top-left corner, indicating high true positive rate with low false positives.
- ✓ The PR curve shows consistently high precision across a wide range of recall, meaning it identifies minority instances with fewer errors.

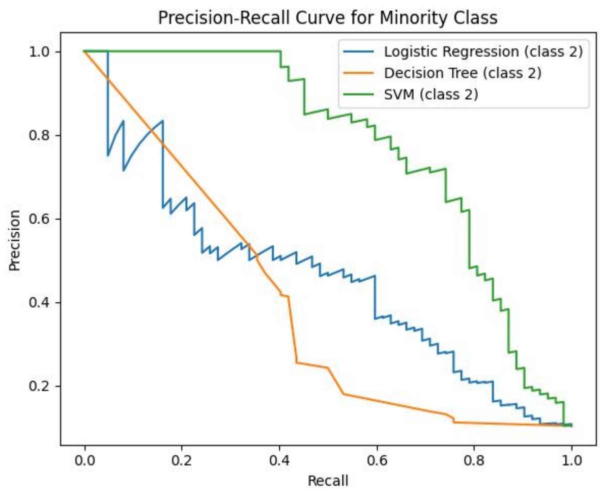
**SVM** performs well too, with smooth curves and solid separation, but slightly trails Logistic Regression in precision at higher recall levels.

- ✓ It still maintains good balance and is robust to class imbalance due to its margin-based decision boundary and probability calibration.

**Decision Tree** struggles. Its ROC curve is jagged and lower, and its PR curve drops sharply, shows poor generalization and high false positive rates for the minority class.

**Conclusion:** Logistic Regression handles imbalance best, followed by SVM. Decision Tree requires techniques like "class-weight balancing".


ROC Curve for Minority Class
TPR
FPR
Logistic Regression (class 2)
Decision Tree (class 2)
SVM (class 2)


Precision-Recall Curve for Minority Class
Precision
Recall
Logistic Regression (class 2)
Decision Tree (class 2)
SVM (class 2)