Module 06: Classification — Cheat Sheet

ML for Innovation Research — Prof. J. Osterrieder

Logistic Regression

Models the probability of class 1:

$$P(y{=}1 \mid \mathbf{x}) = \sigma(\beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p)$$

where $\sigma(z) = \frac{1}{1+e^{-z}}$ (sigmoid function).

- Linear decision boundary.
- Coefficients are interpretable (log-odds).
- Good baseline; works well with few features.

Random Forest

- Ensemble of B decision trees (bagging).
- Each tree trained on a bootstrap sample.
- At each split, consider a random subset of features.
- Final prediction: **majority vote** (classification).
- Handles non-linear relationships, robust to outliers.
- Feature importance available via `feature_importances_`.

Confusion Matrix

	Predicted +	Predicted −
Actual +	TP	FN
Actual −	FP	TN

Classification Metrics

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP} \qquad \text{Recall} = \frac{TP}{TP + FN}$$

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Metric	Use when…
Accuracy	Classes are balanced
Precision	False positives are costly
Recall	False negatives are costly
F1	You need a single balanced metric
AUC	Comparing models across thresholds

ROC Curve & AUC

- Plots True Positive Rate vs. False Positive Rate at all thresholds.
- **AUC** = area under the ROC curve.
- AUC = 0.5 → random; AUC = 1.0 → perfect.
- Threshold-independent: good for model comparison.

Key Code Snippets

```
from sklearn.ensemble import \
    RandomForestClassifier
from sklearn.linear_model import \
    LogisticRegression
from sklearn.model_selection import \
    cross_val_score
from sklearn.metrics import (
    classification_report,
    roc_auc_score,
    confusion_matrix)

# Random Forest
rf = RandomForestClassifier(
    n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)

# Cross-validation
scores = cross_val_score(
    rf, X, y, cv=5, scoring='f1')

# Evaluation
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
auc = roc_auc_score(y_test,
    rf.predict_proba(X_test)[:, 1])
```

Practical Tips

- Always check class balance *first*.
- Use `stratify=y` in train/test split.
- For imbalanced data: set `class_weight='balanced'` or use SMOTE.
- Report multiple metrics, not just accuracy.