IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

# Modeling

Data Science Toolkit & Applications

Lina Fahed & Laurent Brisson

## Modeling: Building Intelligent Systems from Data
## The Heart of Data Science

From Algorithms to **Predictions & Insights** in CRISP-DM

**Core Objective:**

- Transform prepared data into predictions, classifications, or insights.

Example:

- Predicting customer churn
- classifying spam emails
- forecasting sales.

"All models are wrong, but some are useful." — George Box

# Where Does Modeling Fit?

**CRISP-DM Steps:**

1. Business Understanding ✅
2. Data Understanding ✅
3. Data Preparation ✅
4. **Modeling** (Current focus) 🎯
5. Evaluation
6. Deployment

Modeling Goal: **Select and apply algorithms to solve the business problem.**

# What Does Modeling Involve?

1. **Select Modeling Technique**:
    - Choose between classification, regression, clustering, etc. 🎯
2. **Split Data**:
    - Train/validation/test sets
    - e.g., 70%/15%/15% 🔀
3. **Train Model:**
    - Fit the algorithm to the training data. 🏋️
4. **Tune Hyperparameters**:
    - Optimize model performance
    - e.g., grid search ⚙️

# Choosing the Right Tool for the Job 🛠️

| Type | Use Case | Example Algorithms |
|---|---|---|
| Classification | Predict categories (e.g., spam/not spam) | Logistic Regression, Random Forest, SVM |
| Regression | Predict continuous values (e.g., price) | Linear Regression, XGBoost, Neural Networks |
| Clustering | Group similar data points | K-Means, DBSCAN, Hierarchical |
| Association | Find relationships (e.g., market basket) | Apriori, FP-Growth |
| Time Series Forecasting | Predict future values (e.g., sales) | ARIMA, LSTM, Prophet |

# How to Choose an Algorithm? 🤔

Factors to consider

**Problem Type**

- Classification? Regression? Clustering?

**Data Size**

- Small datasets → Simpler models (e.g., Logistic Regression).
- Large datasets → Scalable models (e.g., XGBoost, Neural Networks).

**Interpretability**

- Need explanations? → Use Linear Regression or Decision Trees.

**Performance**:

- Need high accuracy? → Try Ensemble Methods (e.g., Random Forest).

**Example:**

- Small dataset, need interpretability → Decision Tree.
- Large dataset, high accuracy → XGBoost.

**Trade-off**

- e.g., accuracy vs. interpretability
- there's **no 'best' algorithm**—it depends on the problem.

# Splitting Data for Reliable Models 🔀

**Why** Split?

- Avoid overfitting
- model memorizes training data but fails on new data

**Common Splits:**

- 70% Train / 15% Validation / 15% Test ⇒ for medium-sized datasets.
- 80% Train / 20% Test ⇒ for larger datasets.

# Fitting the Model to Data 🏋️

**Steps:**

1. Initialize Model: Choose algorithm, e.g., RandomForestClassifier().
2. Fit to Training Data: `model.fit(X_train, y_train)`
3. Predict on Validation Data: `model.predict(X_val)`

python

```python
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
model.fit(X_train, y_train)
predictions = model.predict(X_val)
```

# Hyperparameters Fine-Tuning for Better Performance ⚙️

**What are Hyperparameters?**

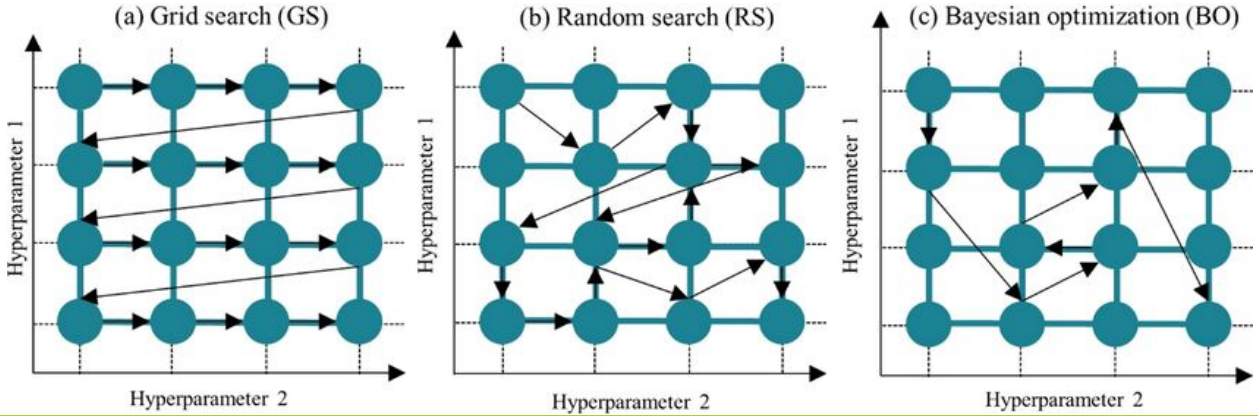
- Settings for the model
- e.g., max_depth in Decision Trees
- e.g., n_estimators in Random Forest

Example:

- Tune max_depth and min_samples_split for a Decision Tree.

**Tuning Methods:**

- **Grid Search**: Test **all** combinations of hyperparameters.
- **Random Search**: Test **random** combinations (faster for large spaces).
- **Bayesian Optimization**: Smarter search using **probability**.



# How Good Is Your Model? 📊

Metrics for **Classification**:

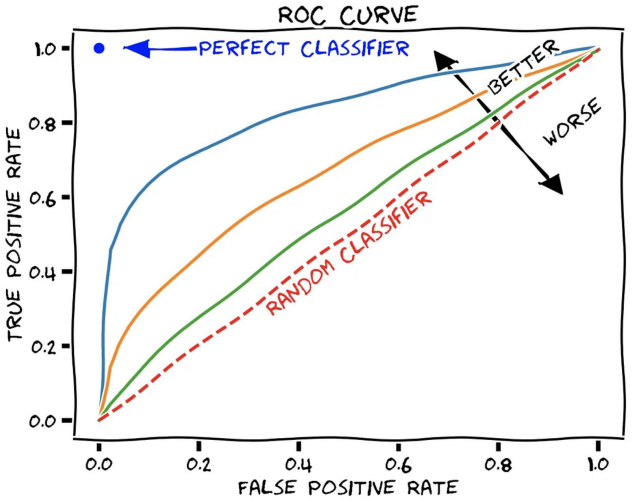
- **Accuracy**: % of correct predictions.
- **Precision**: % of true positives among predicted positives.
- **Recall**: % of true positives among actual positives.
- **F1-Score**: Harmonic mean of precision and recall.
- **ROC-AUC**: Model's ability to distinguish classes.

Metrics for **Regression**:

- **MAE** (Mean Absolute Error): Average absolute error.
- **MSE** (Mean Squared Error): Average squared error.
- **$R^2$:** % of variance explained by the model.

## Confusion Matrix

| | Actually Positive (1) | Actually Negative (0) |
|---|---|---|
| Predicted Positive (1) | True Positives (TPs) | False Positives (FPs) |
| Predicted Negative (0) | False Negatives (FNs) | True Negatives (TNs) |



# Avoid These Modeling Mistakes! ⚠️

| Pitfall | Example | Solution |
|---|---|---|
| Overfitting | Model performs well on training data but poorly on test data. | Use regularization, cross-validation, or more data. |
| Underfitting | Model performs poorly on both training and test data. | Use a more complex model or better features. |
| Data Leakage | Test data influences training (e.g., scaling after split). | Split data before any preprocessing. |
| Ignoring Class Imbalance | Model biased toward majority class. | Use resampling (SMOTE) or class weights. |
| Not Tuning Hyperparameters | Using default hyperparameters. | Always tune hyperparameters (e.g., GridSearchCV). |

# Essential Tools and Libraries 🛠️

Python Libraries:

- Scikit-learn (classic ML algorithms)
- XGBoost/LightGBM (gradient boosting)
- TensorFlow/PyTorch (deep learning)
- Statsmodels (statistical models)

# What to Remember

- **Modeling is Iterative**
  - You'll often revisit Data Preparation
  - You'll often try new algorithms 🔄
- **No Free Lunch**
  - No single algorithm works best for all problems
  - Test multiple models! 🎯
- **Evaluate Properly**
  - Use the right metrics
  - avoid data leakage 📊