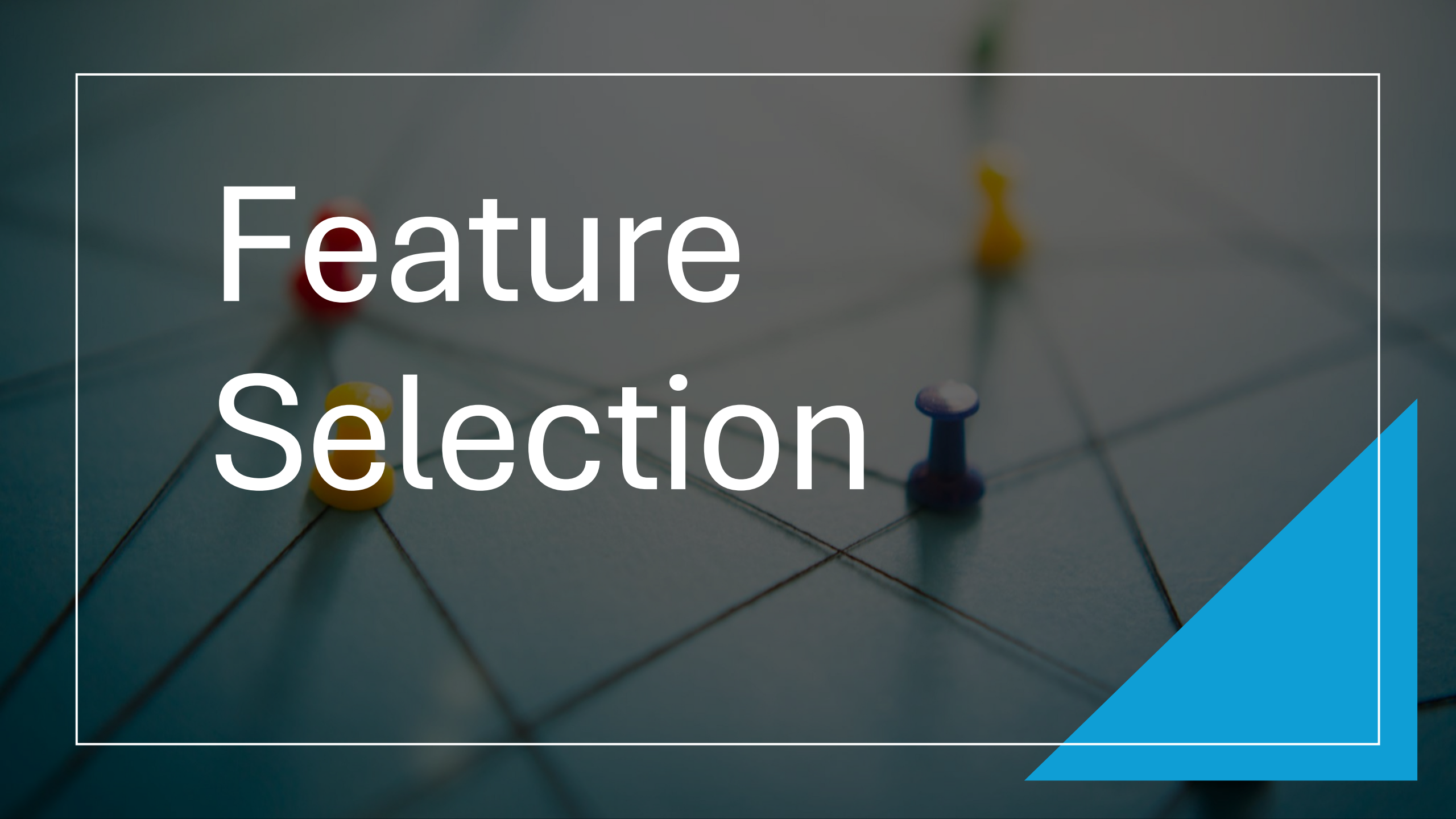

Feature
Selection

What We Learned Yesterday

✅ **Machine Learning Basics**: Supervised vs unsupervised learning

✅ **Decision Trees**: How they work and make decisions

✅ **Model Evaluation**: Accuracy, confusion matrix, business impact

✅ **Feature Importance**: Understanding which factors matter most

Learning Objectives

By the end of this session, you will be able to:

- **Understand** the concept of feature selection and why it's crucial in banking
- **Explain** how the Boruta algorithm works using simple analogies
- **Compare** different feature selection methods (Filter, Wrapper, Embedded)
- **Apply** Boruta to automatically select the most relevant features
- **Evaluate** the impact of feature selection on model performance
- **Implement** a complete feature selection pipeline for credit risk assessment

🍕 Pizza Restaurant Analogy

Imagine you own a pizza restaurant and want to predict which customers will order again:

All Possible Information (Features):

Customer's age, gender, favorite color, shoe size, pet's name, mother's maiden name, favorite movie, zodiac sign, what they had for breakfast, number of siblings, favorite sport, car model, favorite pizza topping, how much they spent, how long they waited, restaurant rating they gave, day of week, weather outside, their mood, etc.

THE PROBLEM:

SOME INFORMATION IS **RELEVANT** (PIZZA TOPPING PREFERENCE, AMOUNT SPENT, RATING GIVEN)

SOME INFORMATION IS **IRRELEVANT** (SHOE SIZE, PET'S NAME, ZODIAC SIGN)

SOME INFORMATION IS **NOISE** (RANDOM FACTORS THAT DON'T REALLY MATTER)

Why Feature Selection Matters

1. The Curse of Dimensionality

Imagine trying to find your friend in a playground (2D), then in a building (3D), then in a city, then in a country. The more dimensions you add, the harder it becomes to find patterns!

Banking Example:

- 10 features: Easy to analyze customer patterns
- 100 features: Harder to identify what really drives default risk
- 1000 features: Nearly impossible to separate signal from noise

Why Feature Selection Matters

2. Computational Efficiency ⚡

- **Fewer features** = Faster loan decisions
- **Relevant features** = More accurate predictions
- **Less storage** = Lower infrastructure costs

3. Model Interpretability 🔍

- **Regulatory Requirement:** Banks must explain loan decisions to customers and regulators. A model with 5 key features is much easier to explain than one with 50 features.

4. Avoiding Overfitting 🎯

- **Simple Analogy:** Like memorizing exact exam questions vs. understanding concepts. Models with too many features might memorize training data but fail on new customers.

Types of Feature Selection Methods

1. Filter Methods

- **Like a Coffee Filter**

How it works:

- Examines features individually
- Uses statistical tests to rank features
- Independent of machine learning algorithm

Types of Feature Selection Methods

Examples:

- **Correlation**: How closely related is each feature to loan default?
- **Chi-Square Test**: Which categorical features best distinguish good vs bad customers?
- **Mutual Information**: How much information does each feature provide?

Feature Analysis:
– Credit Score: High correlation with default (keep)
– Loan Amount: Medium correlation with default (keep)
– Favorite Color: No correlation with default (remove)
– Number of Pets: No correlation with default (remove)

Types of Feature Selection Methods

Filter Methods
Pros:

- ✅ Fast and simple
- ✅ Model-independent
- ✅ Good for removing obviously irrelevant features

Cons:

- ❌ Ignores feature interactions
- ❌ Doesn't consider how features work together
- ❌ May miss important combinations

Types of Feature Selection Methods

2. Wrapper Methods

Like Gift Wrapping - Considers the Whole Package

How it works:

- Uses a specific machine learning algorithm
- Tests different feature combinations
- Selects features that improve model performance

Examples:

- **Forward Selection**: Start with no features, add one at a time
- **Backward Elimination**: Start with all features, remove one at a time
- **Recursive Feature Elimination (RFE)**: Systematically eliminate least important features

Types of Feature Selection Methods

Process:
1. Start with: *[Credit Score, Income, Age, Employment]*
2. Test model with all 4 features: 85% accuracy
3. Remove Age: 86% accuracy (Age was hurting performance!)
4. Add Debt-to-Income ratio: 89% accuracy (Good addition!)

Types of Feature Selection Methods

Wrapper Methods

Pros:

- ✅ Considers feature interactions
- ✅ Optimized for specific algorithm
- ✅ Usually better performance

Cons:

- ❌ Computationally expensive
- ❌ Risk of overfitting
- ❌ Results depend on chosen algorithm

Types of Feature Selection Methods

3. Embedded Methods

Like Built-in Quality Control

How it works:

- Feature selection is built into the algorithm
- Model training and feature selection happen simultaneously
- Algorithm automatically identifies important features

Types of Feature Selection Methods

Examples:

- **LASSO Regression**: Automatically shrinks unimportant features to zero
- **Decision Trees**: Naturally select most informative features for splits
- **Random Forest Feature Importance**: Built-in ranking of feature relevance

LASSO Results:

- - Credit Score coefficient: 0.85 (important)
- - Income coefficient: 0.42 (important)
- - Age coefficient: 0.00 (automatically eliminated)
- - Employment coefficient: 0.23 (moderately important)

Types of Feature Selection Methods

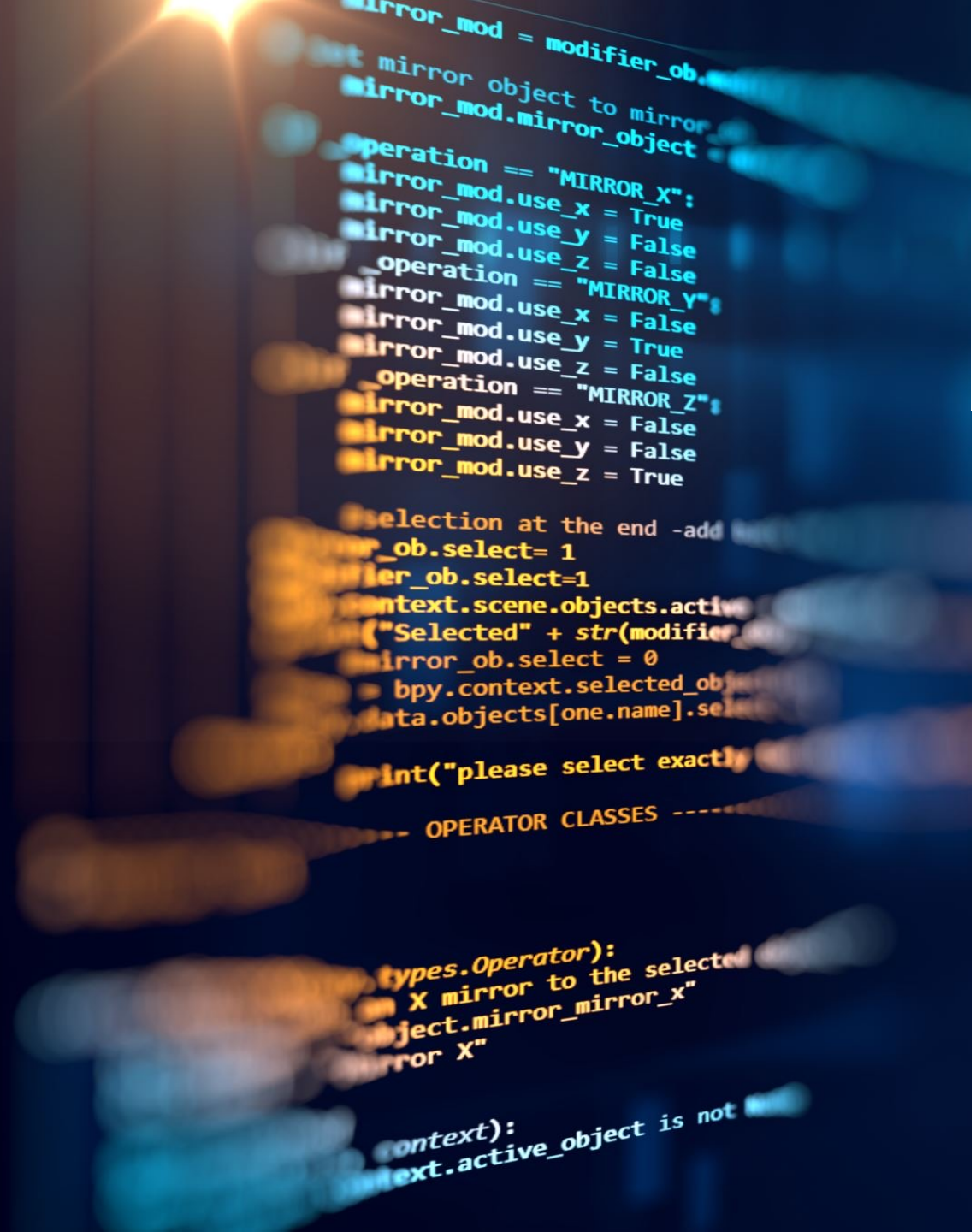
Embedded Methods

Pros:

- ✅ Efficient (combines training and selection)
- ✅ Considers feature interactions
- ✅ Less prone to overfitting than wrapper methods

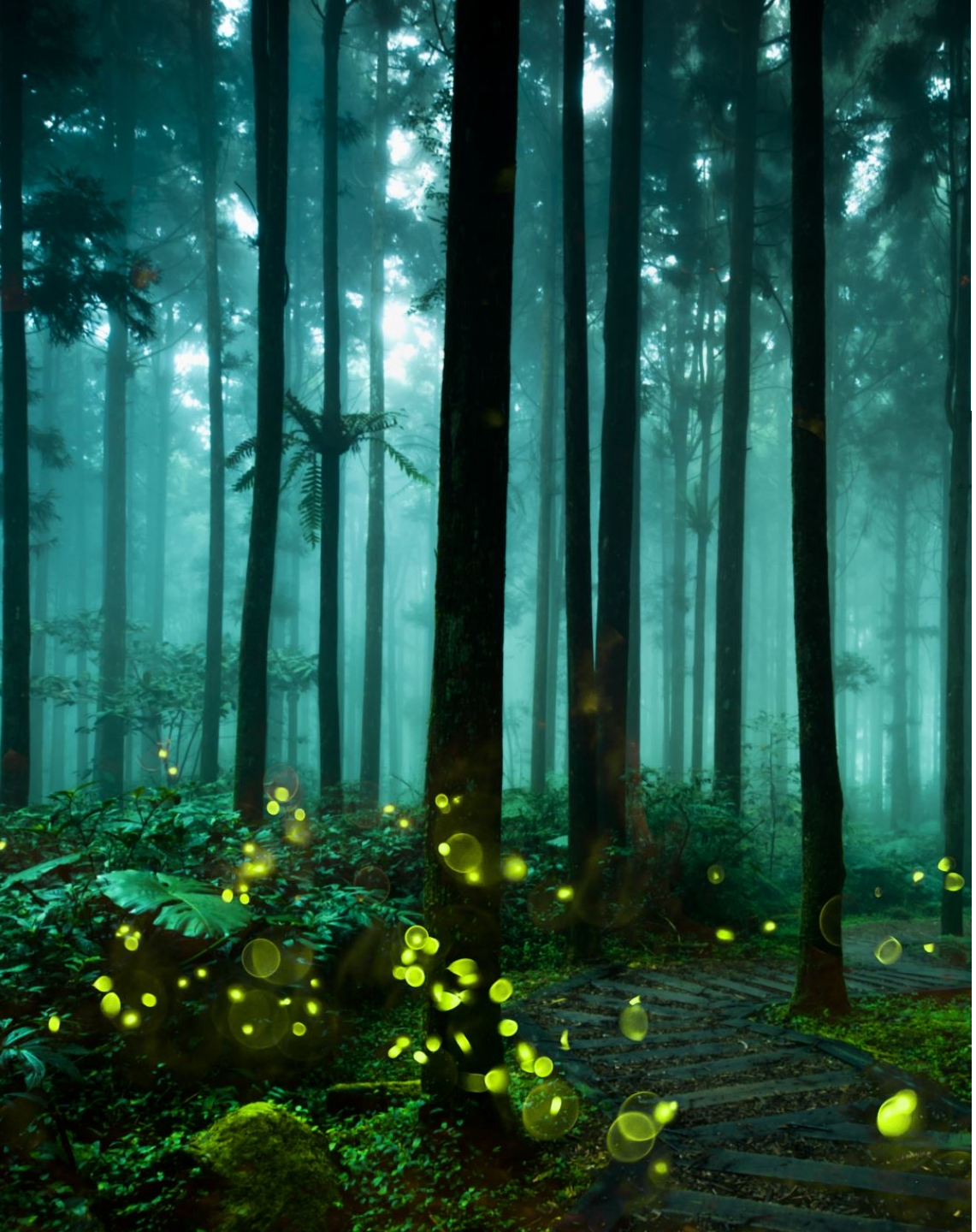
Cons:

- ❌ Algorithm-specific results
- ❌ Limited to algorithms that support it
- ❌ Less control over selection process



Code Along

https://bit.ly/uob2025



What is Boruta?

Name Origin: Boruta comes from Slavic mythology - the guardian spirit of the forest. Just as this spirit protects the forest and knows every tree that belongs, the Boruta algorithm protects your model by identifying which features truly belong!

Boruta's Philosophy: "All-Relevant" Feature Selection

Traditional Approaches (Minimal Optimal):

- Find the **smallest set** of features that gives good performance
- Like finding the minimum number of ingredients for an "okay" pizza

Boruta Approach (All-Relevant):

- Find **ALL features** that contain useful information
- Like identifying every ingredient that makes the pizza better

How Boruta Works

Step 1: Create Shadow Features 👥

Imagine you want to know if your friends are really fast runners. You create "fake friends" by mixing up everyone's running times randomly. Now you can compare: "Is Alice really fast, or just faster than random chance?"

Technical Process:

- Take your original features
- Create "shadow copies" by randomly shuffling each feature's values
- Shadow features represent pure randomness/noise

Original Features:

- Customer A: Credit Score = 750, Income = $50K
- Customer B: Credit Score = 650, Income = $80K
- Customer C: Credit Score = 700, Income = $60K

Shadow Features (randomly shuffled):

- Customer A: Shadow_Credit_Score = 650, Shadow_Income = $60K
- Customer B: Shadow_Credit_Score = 700, Shadow_Income = $50K
- Customer C: Shadow_Credit_Score = 750, Shadow_Income = $80K

Step 2: Train Random Forest on Extended Dataset 🌲

What happens:

- Combine original + shadow features
- Train Random Forest on this extended dataset
- Calculate feature importance for ALL features

- **The Test:** If an original feature is truly important, it should perform better than its random shadow copy!

Step 3: Statistical Testing 📊

- **The Question:** "Is this feature significantly better than random noise?"

Process:

- Calculate Z-scores for each feature
- Compare original features to their shadow copies
- Use statistical tests to determine significance

Step 4: Feature Classification 🏷️

Three Categories:

- **Confirmed Important** ✅
  - Consistently performs better than shadow features
  - Clearly contains useful information
- **Confirmed Unimportant** ❌
  - Performs worse than or similar to shadow features
  - Likely just noise
- **Tentative** ❓
  - Sometimes better, sometimes not
  - Needs more iterations to decide

Step 5: Iterate Until Convergence 🔄

Repeat the process until:

- All features are classified as important or unimportant
- Maximum iterations reached
- No more changes occur

Boruta vs. Traditional Random Forest Feature Importance

Traditional Random Forest Importance

Feature Importance Scores:

- Credit Score: 0.35
- Income: 0.28
- Age: 0.15
- Employment: 0.22

Question: Is 0.15 for Age "important enough"?

Answer: We don't know! No statistical significance test.

Boruta Enhancement

Statistical Significance:

- Credit Score: $Z=15.2$, $p<0.001$ (IMPORTANT)
- Income: $Z=12.8$, $p<0.001$ (IMPORTANT)
- Age: $Z=2.1$, $p=0.45$ (NOT IMPORTANT)
- Employment: $Z=8.5$, $p<0.01$ (IMPORTANT)

Clear Answer: Age is not statistically significant!

Advantages of Boruta

✅ **Comprehensive Feature Discovery**

- Finds ALL relevant features, not just the minimum set
- Important for understanding complete business drivers

✅ **Statistical Rigor**

- Uses proper statistical testing for significance
- Reduces false discoveries through multiple testing correction

✅ **Handles Feature Interactions**

- Random Forest base captures complex relationships
- Identifies features important in combination

✅ **Robust to Noise**

- Explicit comparison with random features
- Strong resistance to selecting irrelevant features

✅ **Reproducible Results**

- Statistical foundation provides consistent outcomes
- Less dependent on random initialization

Limitations of Boruta

❌ **Computational Cost**
- Can be slow with large datasets
- Multiple Random Forest training iterations

❌ **Random Forest Dependency**
- Results tied to Random Forest assumptions
- May miss patterns that other algorithms would catch

❌ **Data Requirements**
- Needs sufficient data for statistical testing
- Requires handling of missing values beforehand

❌ **Categorical Feature Handling**
- May need preprocessing for high-cardinality categories
- Encoding choices can affect results

Boruta Parameters Understanding

n_estimators (Number of Trees)

```
# More trees = more stable but slower
boruta = BorutaPy(
    RandomForestClassifier(n_estimators=100),  # Default
    n_estimators='auto',  # Let Boruta decide
    random_state=42
)
```

Boruta Parameters Understanding

max_iter (Maximum Iterations)

```
# How many rounds of testing
boruta = BorutaPy(
    classifier,
    max_iter=100,  # Stop after 100 rounds if not converged
    random_state=42
)
```

Boruta Parameters Understanding

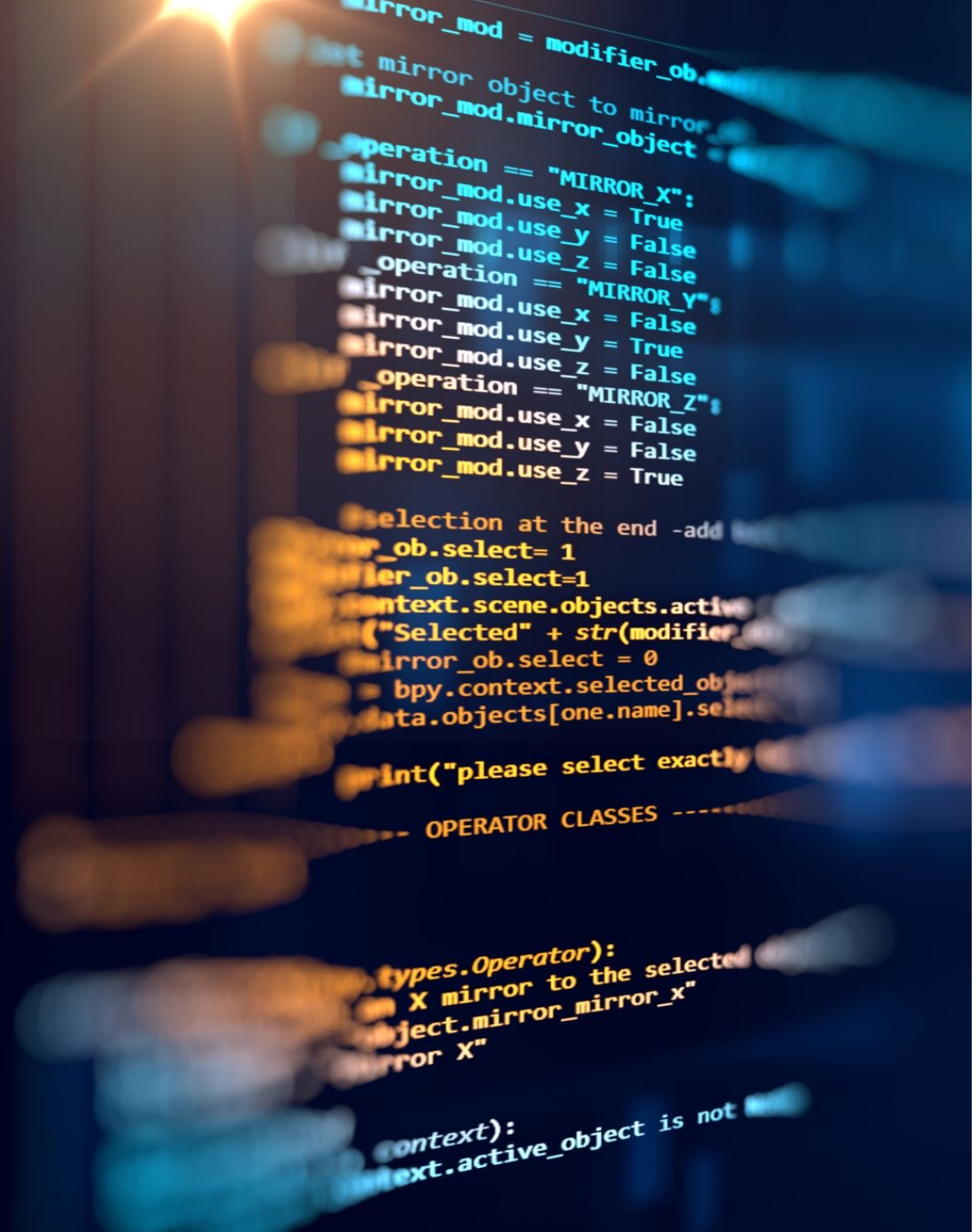
perc (Percentile Threshold)

```
# How strict the comparison
boruta = BorutaPy(
    classifier,
    perc=100,  # Compare to maximum shadow importance (strict)
    perc=90,   # Compare to 90th percentile (more lenient)
    random_state=42
)
```

Boruta Parameters Understanding

alpha (Significance Level)

```
# Statistical significance threshold
boruta = BorutaPy(
    classifier,
    alpha=0.05,  # 5% significance level (standard)
    alpha=0.01,  # 1% significance level (more strict)
    random_state=42
)
```



Code Along

https://bit.ly/uob2025