What is Feature Selection?

- **Definition:** Process of selecting a subset of features most relevant to the target
- **Benefits:**
    - Mitigates computational burden
    - Helps avoid overfitting
    - Enhances model interpretability
- **Particularly important:** When dealing with high-dimensional datasets

Feature Selection in MoDeVa

The `DataSet` class in MoDeVa implements three feature selection strategies:

- `DataSet.feature_select_corr`
  - Selects features based on correlation with target variable
- `DataSet.feature_select_xgbpfi`
  - Selects important features using XGBoost model and permutation importance
- `DataSet.feature_select_rcit`
  - Selects causal features using conditional independence testing

Correlation Coefficient Approach

`DataSet.feature_select_corr` selects features based on:

- Strength of correlation between features and target variable
- Different correlation measures based on variable types:
  - **Pearson's correlation:** For numerical-numerical pairs
  - **Theil's U:** For categorical-categorical pairs
  - **Correlation ratio:** For mixed numerical-categorical pairs
- Features with correlation strength above specified threshold are selected

Implementing Correlation-Based Selection

Basic Usage

```
# Import necessary libraries
from modeva import DataSet

# Load your dataset
dataset = DataSet(data)

# Select features based on correlation with target
selected_features = dataset.feature_select_corr(
    target='target_variable',
    threshold=0.3  # Correlation strength threshold
)

print(f"Selected features: {selected_features}")
```

Correlation Measures in Detail

Pearson's Correlation

- Measures linear relationship
- Range: $[-1, 1]$
- $+1$: Perfect positive correlation
- -1: Perfect negative correlation
- 0: No linear correlation

Theil's U

- Measures association between categorical variables
- Based on information theory
- Range: $[0, 1]$
- Asymmetric measure
- Handles non-linear relationships

Correlation Ratio

- Measures association between numerical and categorical variables
- Ratio of between-group variance to total variance
- Range: $[0, 1]$
- Higher values indicate stronger association

XGBoost with Permutation Feature Importance

`DataSet.feature_select_xgbpfi` approach:

- **Two-step process:**
  1. Fit an XGBoost model using all features and the target
  2. Apply permutation feature importance analysis
- **Selection methodology:**
  1. Obtain importance score for each feature
  2. Sort features by importance (descending order)
  3. Normalize scores to sum to 1
  4. Select top features with accumulated importance above threshold
- **Caution:** Results may be affected if the fitted XGBoost model overfits or underfits

Implementing Feature Importance-Based Selection

Basic Usage

```
# Select features based on XGBoost permutation importance
selected_features = dataset.feature_select_xgbpfi(
    target='target_variable',
    threshold=0.8,  # Accumulated importance threshold
    n_estimators=100  # Number of trees in XGBoost model
)

print(f"Selected features: {selected_features}")
```

Benefits and Limitations

Benefits:

- Captures non-linear relationships between features and target
- Accounts for feature interactions
- Can handle mixed data types (numerical and categorical)
- More robust than simple correlation for complex relationships

Limitations:

- Dependent on XGBoost model quality
- May select redundant features (correlated with each other)
- Results may vary with different random seeds
- Computationally more intensive than correlation-based methods

RCIT and FBEDk Algorithm

`DataSet.feature_select_rcit` implements a two-stage process:

- Combines **Randomized Conditional Independence Test (RCIT)** with **Forward-Backward-Elimination with Early Dropping (FBEDk)**
- **Process:**
  1. Forward selection to identify potentially important features
  2. Backward elimination to remove redundant features
- Uses random Fourier features for non-parametric conditional independence testing
- Capable of identifying features causally related to the target

RCIT: Randomized Conditional Independence Test

RCIT tests whether a feature X is independent of the response Y, given a Markov boundary set Z:

- Notation: $X \perp Y \mid Z$ (X is independent of Y given Z)
- **Process:**
  1. Transform X, Y, Z using random Fourier features
  2. Test null hypothesis: $\Sigma_{XY|Z} = \Sigma_{XY} - \Sigma_{YZ}\Sigma_{ZZ}^{-1}\Sigma_{XZ} = 0$
  3. Compute test statistic: $\|\hat{\Sigma}_{XY|Z}\|_F^2 = n\hat{\Sigma}_{XY} - \hat{\Sigma}_{YZ}(\hat{\Sigma}_{ZZ} + \gamma I)^{-1}\hat{\Sigma}_{XZ}$
  4. Approximate distribution: $\sum_{i=1} \lambda_i z_i^2$, where z_i are i.i.d. standard Gaussian
- Lindsay-Pilla-Basak method approximates CDF under null using Gamma distribution mixture

FBEDk Algorithm: Forward Selection

Forward Selection phase:

1. Initialize with a predefined Markov boundary set
2. Initialize all remaining covariates as candidate features
3. For each candidate feature:
   - Run RCIT test between candidate and response, conditional on Markov boundary
   - Select features with p-value $\leq$ threshold as candidates
4. Add most significant candidate to Markov boundary set
5. Repeat until candidate set is empty
6. Repeat entire forward phase k times (default: $k = 2$)

FBEDk Algorithm: Backward Elimination

Backward Elimination phase:

1. For each feature j in the Markov boundary set:
   - Temporarily remove feature j from the set
   - Run RCIT test between feature j and response Y, conditional on temporary Markov boundary
   - If p-value $>$ threshold, permanently remove feature j
2. Resulting Markov boundary contains only causally significant features

Implementing Conditional Independence-Based Selection

Basic Usage

```
# Select features using RCIT and FBEDk
selected_features = dataset.feature_select_rcit(
    target='target_variable',
    threshold=0.05,  # Significance level
    n_forward_phase=2  # Number of forward phases
)

print(f"Selected features: {selected_features}")
```

Benefits and Limitations of RCIT-Based Selection

Benefits:

- Can handle non-linear relationships
- Selects features causally related to response
- Reduces redundancy among selected features
- Statistically rigorous approach

Limitations:

- Computationally intensive
- Results may vary with different initial Markov boundary sets
- Requires careful selection of significance threshold
- May require larger sample sizes for reliable results

Comparing Feature Selection Methods

Aspect	**Correlation**	**XGBoost PFI**	**RCIT-FBEDk**
Relationship type	Linear (mostly)	Non-linear	Non-linear
Computational cost	Low	Medium	High
Detects causality	No	Partially	Yes
Handles redundancy	No	No	Yes
Implementation	Simple	Moderate	Complex
Feature interactions	No	Yes	Yes
Stability	High	Medium	Medium

When to Use Each Method

Use Correlation-Based Selection when:

- Dataset is small to medium-sized
- Quick exploratory analysis is needed
- Relationships are primarily linear
- Computational resources are limited

Use XGBoost Feature Importance when:

- Non-linear relationships are expected
- Feature interactions are important
- Predictive performance is the primary goal
- Medium computational resources are available

Use RCIT-FBEDk when:

- Causal features are of interest
- High feature redundancy is expected
- Dataset is medium to large
- Strong statistical guarantees are required

Best Practices for Feature Selection

1. **Start simple:** Begin with correlation-based method for initial exploration
2. **Cross-validate:** Evaluate model performance with selected features
3. **Consider stability:** Apply selection methods with different random seeds
4. **Combine methods:** Use intersection or union of features from different methods
5. **Domain knowledge:** Incorporate domain expertise in final feature selection
6. **Iterative approach:** Refine feature set based on model performance
7. **Monitor overfitting:** Check if reduced feature set improves generalization
8. **Document process:** Keep track of selection criteria and results

Complete Feature Selection Workflow

```
from modeva import DataSet
# Load your dataset
dataset = DataSet(data)
# Apply multiple feature selection methods
corr_features = dataset.feature_select_corr(
    target='target_variable', threshold=0.3)
xgb_features = dataset.feature_select_xgbpfi(
    target='target_variable', threshold=0.8)
causal_features = dataset.feature_select_rcit(
    target='target_variable', threshold=0.05)
# Find common features across methods
common_features = set(corr_features) & set(xgb_features) &
set(causal_features)
# Use selected features for modeling
final_features = list(common_features)
```

Summary

- **Feature selection** improves model performance, interpretability, and efficiency
- **Modeva provides three approaches:**
    - Correlation-based: Simple, efficient for linear relationships
    - XGBoost importance: Captures non-linear relationships and interactions
    - RCIT-FBEDk: Identifies causal features with statistical rigor
- **Method selection** depends on:
    - Dataset characteristics
    - Modeling objectives
    - Available computational resources
    - Desired statistical properties
- **Combined approach** often yields the most robust feature set