Feature Selection

By: **Annamalai A** with **GPT** assist

Feature selection is the process of choosing the most relevant input features for model training.

It helps to:

- Improve model accuracy
- Reduce overfitting
- Reduce training time
- Remove irrelevant or redundant features
- Improve model interpretability

Three broad categories:

1. Filter methods
2. Wrapper methods
3. Embedded methods

Filter Methods

Filter methods use statistical tests to score and select features.

They do not depend on any machine learning model.

Removing Low-Variance Features

- Uses `VarianceThreshold`
- Removes features with low or zero variance
- Zero variance → same value for all samples
- Low-variance features carry little useful information

Univariate Feature Selection

Evaluates each feature individually based on a statistical test.

Methods:

- `SelectKBest` → top k features
- `SelectPercentile` → top p% of features
- `GenericUnivariateSelect` → flexible mode

Common scoring functions:

- `f_classif` (ANOVA F-value) — classification
- `f_regression` — regression
- `chi2` — categorical/non-negative data
- `mutual_info_classif` / `mutual_info_regression` — non-linear dependencies

Important note: Use classification scores only for classification and regression scores only for regression.

Mutual Information, Chi-Square, and F-Statistics

Mutual Information (MI)

- Measures dependency between feature and target
- MI = 0 means independence

Chi-Square Test

- Tests dependence between feature and class label
- Only works with non-negative feature values

F-Statistics (ANOVA F-value)

- Measures linear dependency
- Used in both classification and regression

Wrapper Methods

Wrapper methods use a model to evaluate subsets of features.

They are more accurate but more computationally expensive.

Recursive Feature Elimination (RFE)

Steps:

1. Train a model on all features

2. Compute feature importance
3. Remove the least important feature
4. Repeat until required number of features remain

`RFE` requires you to manually specify the number of features.

RFECV — RFE with Cross-Validation

- Automatically selects the optimal number of features
- Uses cross-validation to evaluate different subsets
- Much more reliable but slower

Select From Model

Selects features based on the feature importance of an estimator.

Key points:

- Works with models that expose `coef_` or `feature_importances_`
- Accepts thresholds such as `"mean"`, `"median"`, `"0.5*mean"`
- Useful for linear models with L1 penalty and tree-based models

Sequential Feature Selection

A greedy approach to selecting features.

Types:

- **Forward Selection** → start with 0 features, add one at a time
- **Backward Selection** → start with all features, remove one at a time

Use backward selection when selecting a large portion of features (fewer iterations).

Drawbacks:

- Slower than RFE and SelectFromModel
- Model must be trained many times

Column Transformer

Allows different transformations for different columns.

Example uses:

- Scale numeric features
- One-hot encode categorical features
- Combine multiple preprocessing steps

A ColumnTransformer entry is written as:

```
(name, transformer, column_indices)
```

All outputs are concatenated into a single matrix.

Transformed Target Regressor

Applies a transformation to the target variable (y) during training.

Useful when:

- Target distribution is skewed
- Log-transform or scaling is required

Steps:

1. Transform y
2. Train the regressor
3. Apply inverse transform to predictions

Dimensionality Reduction — PCA

Principal Component Analysis (PCA) reduces dimensionality by projecting data onto new axes.

Key points:

- Based on Singular Value Decomposition (SVD)
- PC1 captures maximum variance
- Each subsequent component has decreasing variance
- Components are orthogonal

- Choose the first k components to retain most information

Pipelines

Used to chain multiple preprocessing steps and a model.

Benefits:

- Prevents data leakage
- Ensures identical preprocessing in train/test
- Enables joint hyperparameter tuning
- Cleaner and more organized code

Two ways to create pipelines:

- `Pipeline([...])`
- `make_pipeline(...)`

Parameter access pattern:

```
<estimator>__<parameter>
```

Grid Search with Pipelines

Grid search allows tuning:

- Preprocessing steps
- Model type
- Model hyperparameters

All in a single unified pipeline.

Caching Transformer Outputs

Using the `memory` parameter in pipelines:

- Stores intermediate computation
- Speeds up grid search for large datasets

FeatureUnion

FeatureUnion applies multiple transformers in parallel and concatenates outputs.

Used for:

- Combining numeric and categorical transformations
- Combining PCA features with untransformed features
- Mixing multiple feature engineering steps

Difference from Pipeline:

- Pipeline is sequential
- FeatureUnion is parallel