How To Handle Over Fitting

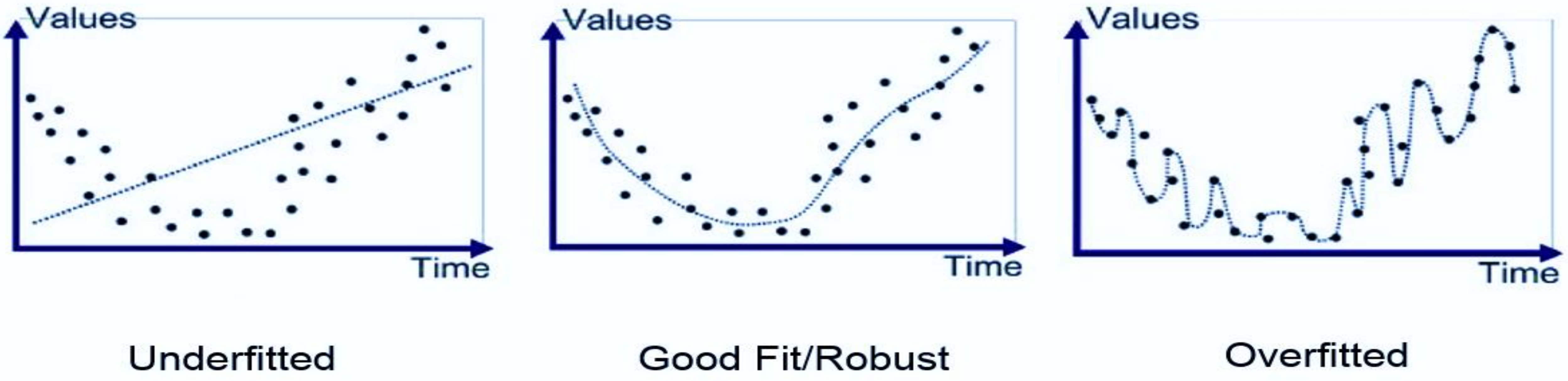
- What is over fitting
- How to recognize it
- How to handle it
  - regularization
  - pruning

Overfitting vs. Underfitting

Overfitting: Memorizes noise, performs poorly on unseen data.

Underfitting: Too simple, misses patterns.

Good Fit: Generalizes well to new data.



Accuracy Score Method

Calculate accuracy on both training and test sets. **Large gap** Indicates overfitting.

```
train_accuracy = accuracy_score(y_train, y_train_pred)
test_accuracy = accuracy_score(y_test, y_test_pred)

print(f"Train Accuracy: {train_accuracy}")
print(f"Test Accuracy: {test_accuracy}")
print(f"Delta:  {train_accuracy-test_accuracy}")
```

```
Train Accuracy: 0.8342696629213483
Test Accuracy: 0.8212290502793296
Delta:  0.013040612642018723
```

Learning Curve Visualization

learning_curve will actualy train the data (not look at past training results)

- **Overfitting:** High train accuracy, low test accuracy.
- **Good Fit:** Train and test accuracy converge.

```
from sklearn.model_selection import learning_curve

train_sizes, train_scores, test_scores = learning_curve(
    dt, X, y, cv=5, scoring='accuracy',
    train_sizes=np.linspace(0.1, 1.0, 10),
    random_state=42
)
```

Cross validation- how many folds

model

How to split the training data to subsets

Techniques to Address Overfitting

Adding Data: increase the amount of data given to the model.

Feature Selection: choosing the more important features, dropping the rest.

Ensemble: increase the randomness thus having more variance.

Early Stopping: Halt training when validation performance degrades.

Regularization: adds constraints or penalties to the model's parameters to control its complexity and prevent overfitting by discouraging overly large coefficients.

Pruning (Decision Trees): Remove low-importance branches.

```python
lr = LogisticRegression(penalty='l1', C=0.01, solver ='liblinear')
```

Penalty-

A term added to the cost function in regularization to discourage large coefficients.

L1 Penalty (Lasso): Adds the absolute value of coefficients ($\lambda\ \Sigma|w|$).

Promotes sparsity (some coefficients = 0).

L2 Penalty (Ridge): Adds the square of coefficients ($\lambda\ \Sigma w^2$).

Promotes smaller, balanced coefficients.

```python
lr = LogisticRegression(penalty='l1', C=0.01, solver ='liblinear')
```

λ Controls the strength of regularization:

- **High λ:** Strong regularization, smaller coefficients.
- **Low λ:** Weak regularization, more complex model.

•Optimal λ: Found via **cross-validation**

C-

An Inverse of λ, stronger regularization.

λ – how "strong" will the penalty be?

$C = \frac{1}{\lambda}$ it is never 0.

If it is small there is a strong penalty for large margins, it keeps the regularization strength

If it is big(10, 100 etc) it reduces the regularization strength.

```python
lr = LogisticRegression(penalty='l1', C=0.01, solver ='liblinear')
```

Solver-

defines the algorithm used for optimization when training a model

Examples:

liblinear: Best for small datasets; supports L1 and L2 regularization.

saga: Efficient for large datasets; supports L1, L2, and ElasticNet.

lbfgs: Ideal for multiclass problems; supports only L2.

Pruning Benefits:

- Reduces overfitting.
- Creates simpler, more interpretable trees.
- Balances tree complexity and generalization.

max_depth: Maximum depth of the tree.

Prevents the tree from growing too deep.

Example: *DecisionTreeClassifier(max_depth=3).*

min_samples_split: Minimum samples needed to split a node.

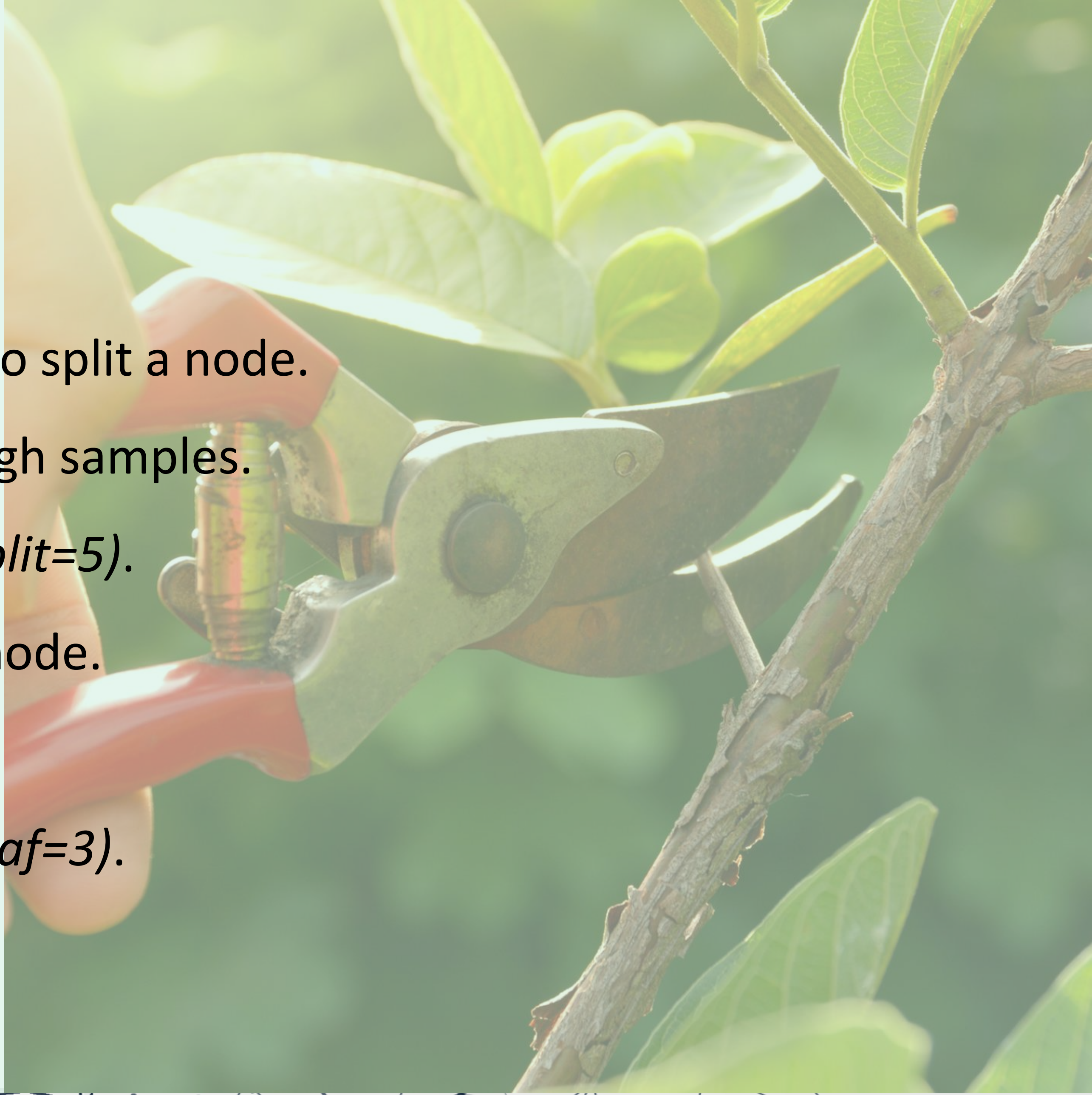
Ensures splits happen only when there are enough samples.

Example: *DecisionTreeClassifier(min_samples_split=5)*.

min_samples_leaf: Minimum samples in a leaf node.

Avoids very small leaf nodes.

Example: *DecisionTreeClassifier(min_samples_leaf=3)*.



Prunning

- **Overfitting:** When a model learns the noise and details of training data too well, leading to poor generalization on unseen data.
- **How to recognize it:** A significant gap between training accuracy (high) and test accuracy (low).
- **How to handle it:** Use techniques like regularization, pruning, or collecting more data to reduce model complexity.

- **Regularization:** Adds a penalty to the cost function to discourage overly complex models.
- **Pruning:** Simplifies decision trees by limiting their growth or removing unnecessary branches.