ENSEMBLE TECHNIQUES: A STRATEGY FOR MODEL OPTIMIZATION


GET READY
START
GET READY

Machine Learning Concepts 1.

1. Bias- Variance Trade off

Bias- Variance Trade off

Error in Machine Learning

In machine learning, an error is a **measure of how accurately an algorithm can make predictions for the previously unknown dataset.**

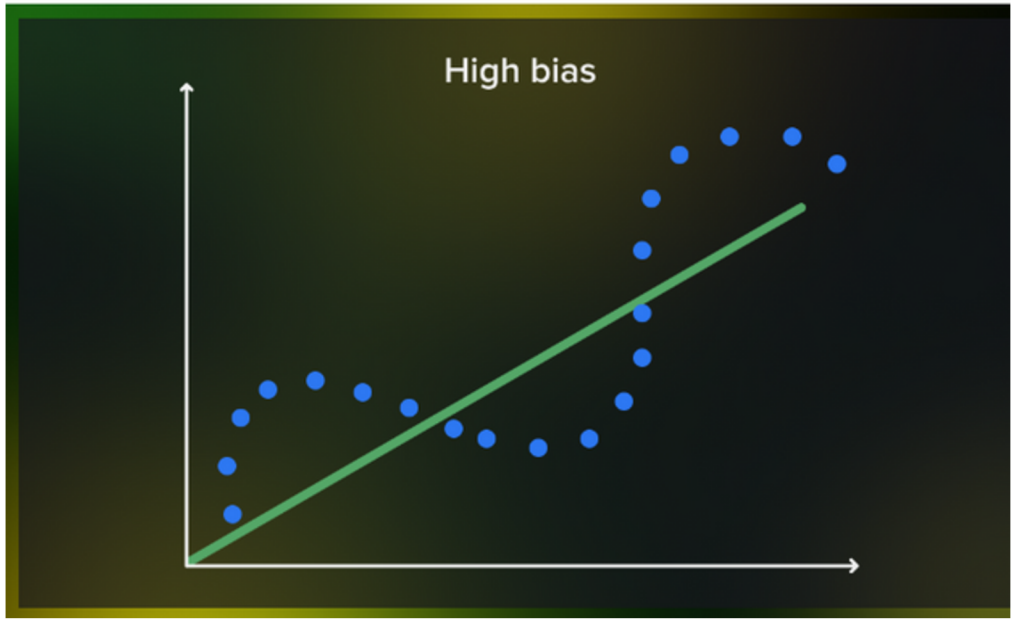
Total Errors in Machine Learning came from 3 Types of Errors

1. Bias
2. Variance
3. Irreducible error

Irreducible error is the one that we can't avoid or possibly eliminate. They are caused by elements outside of our control, such as noise from observations.

What is Bias ?

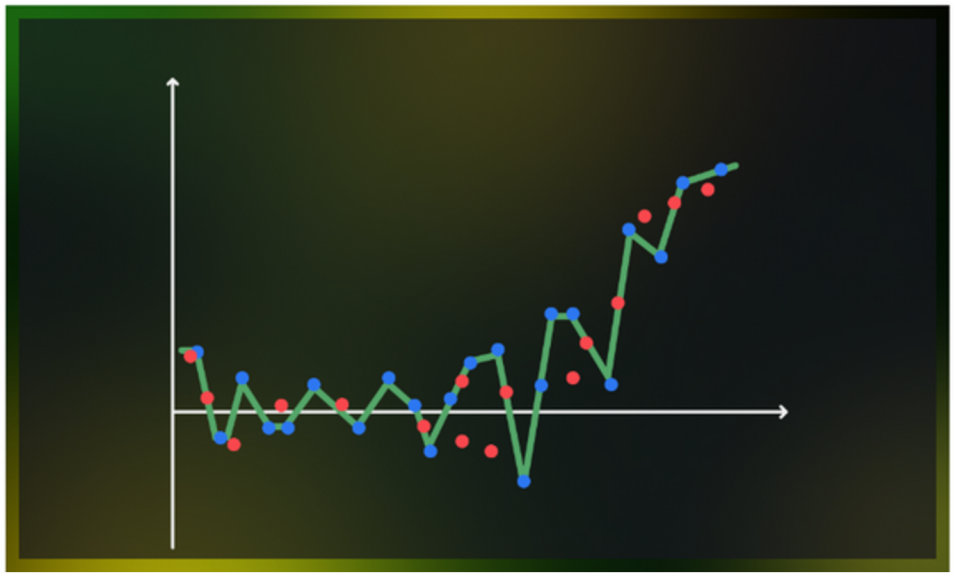
Bias is the difference between the average prediction of our model and the correct value which we are trying to predict.



- Low complexity (simple) models tend to have high bias.
- Bias is typically measured by evaluating the performance of a model on a training dataset
- Model with high bias pays very little attention to the training data and oversimplifies the model.

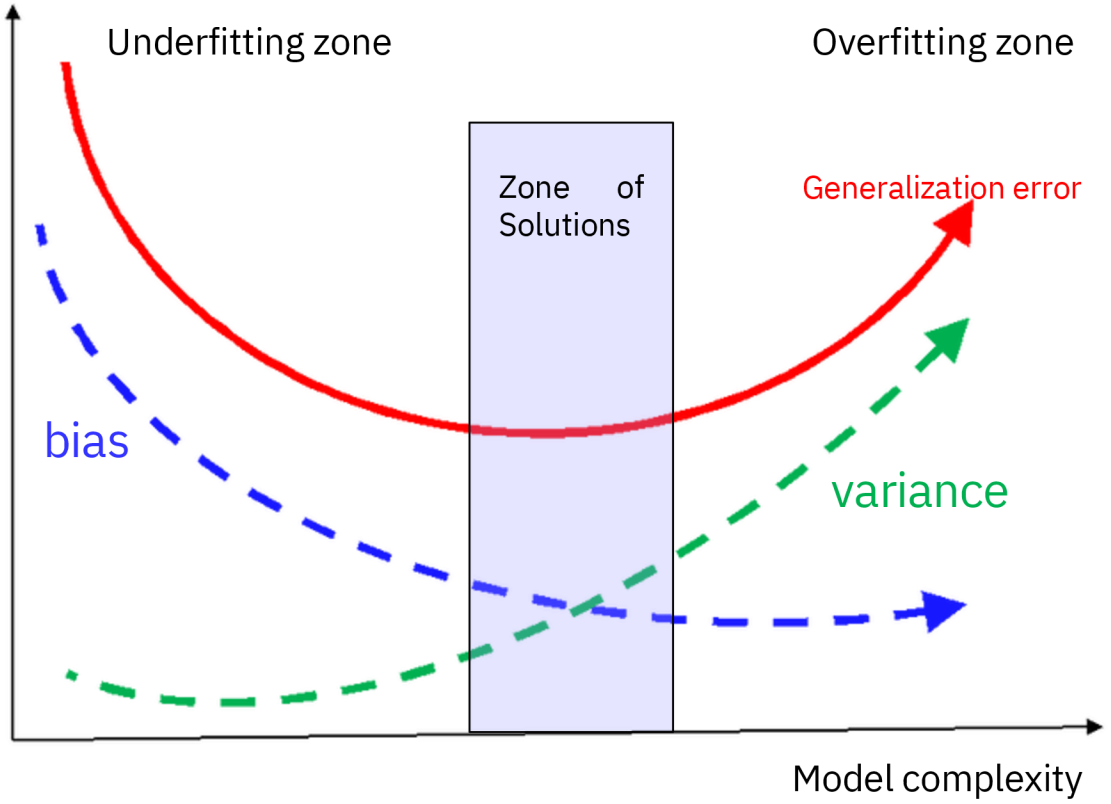
What is Variance?

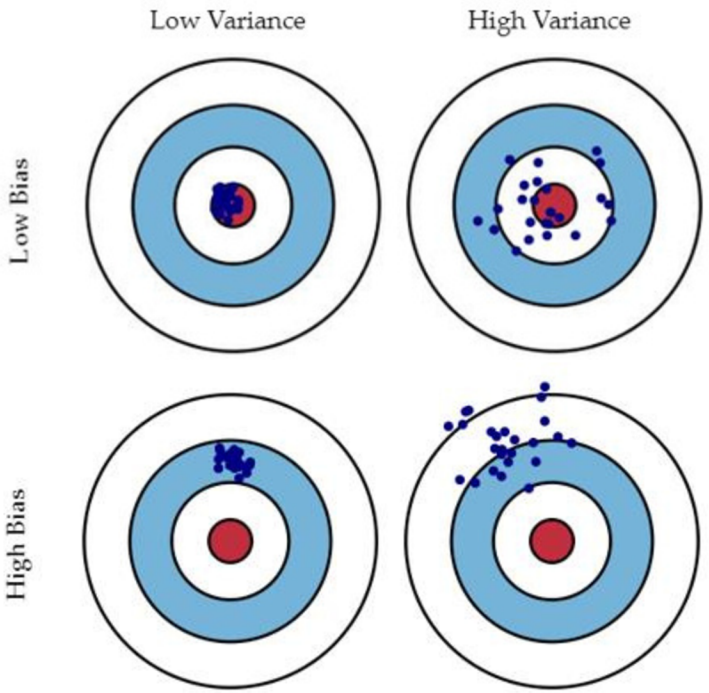
Variance is the variability of model prediction for a given data point or a value which tells us spread of our data. .



- Variance is the **variability in the model prediction, meaning how much the predictions will change if a different training dataset is used**
- A model with a **high level of variance depends heavily on the training dat**a and, consequently, has a limited ability to generalize to new, unseen figures.
- High complexity models tend to have high variance.

Tradeoff between bias and variance



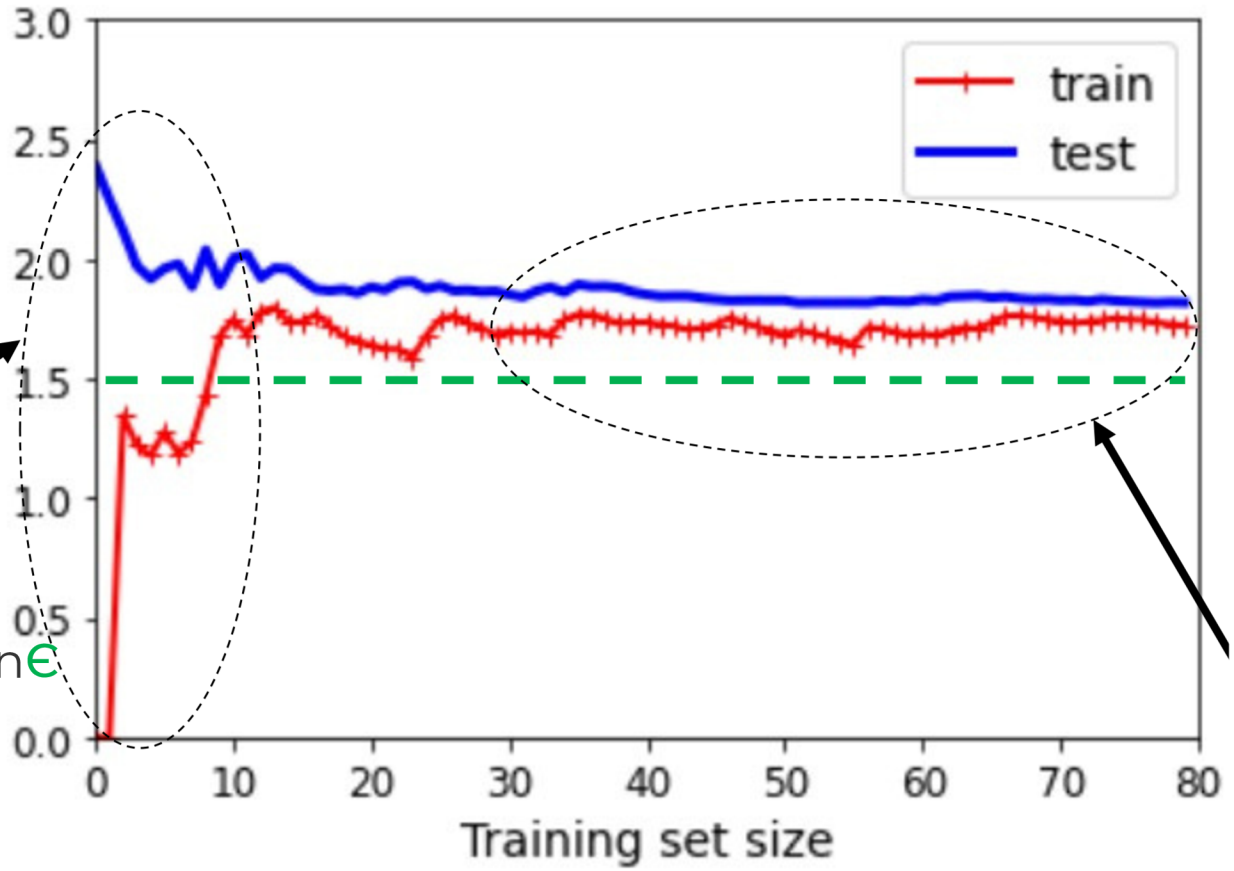


$$Err(x) = \left(E[\hat{f}\,(x)] - f(x)\right)^2 + E\left[\left(\hat{f}\,(x) - E[\hat{f}\,(x)]\right)^2\right] + \sigma_e^2$$

$$Err(x) = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$$

Tradeoff between bias and variance

Learning curves to Detect High Bias and High Variance



Regime 1(High Variance)

Symptoms:

1.Training error is much lower than test error

2.Training error is lower than ϵ

3.Test error is above ϵ

Remedies:

•Add more training data
•Reduce model complexity --complex models are prone to high variance
•Reduce dimensionality of the data

Desired error threshold **(ϵ)**

Regime 2 (High Bias)

Symptoms:

1.Training error is higher than ϵ

Remedies:

•Use more complex model (e.g., kernelize, use non-linear models)
•Add/engineer features with higher predictive power

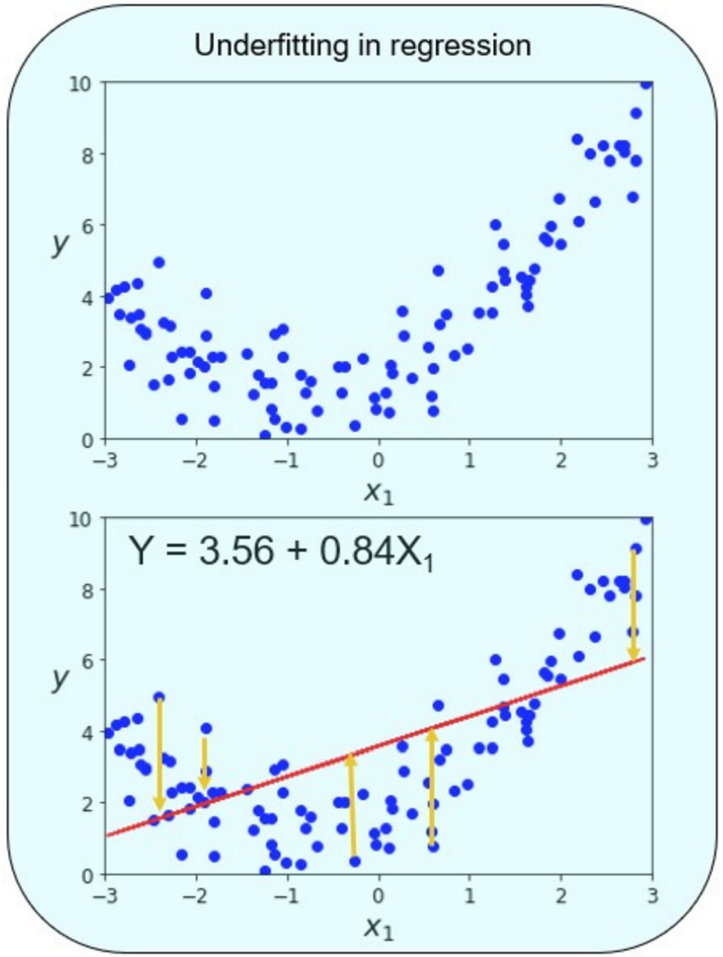
Under Fitting

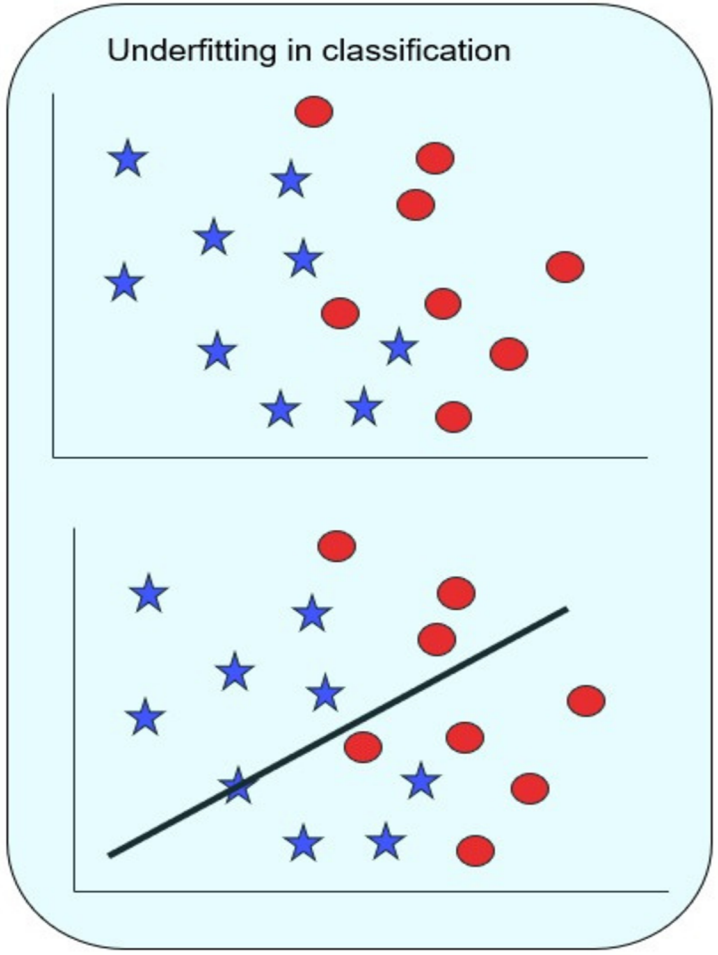
The goal of supervised learning is to build a model on the training dataset that can make accurate predictions on new, unseen data... i.e., ability to generalize from training set to test set, while maintaining high accuracy

Underfitting

When the model is too simple and does not capture the underlying structure of the data.. Characterized by low accuracy.

Too simple to explain variance →

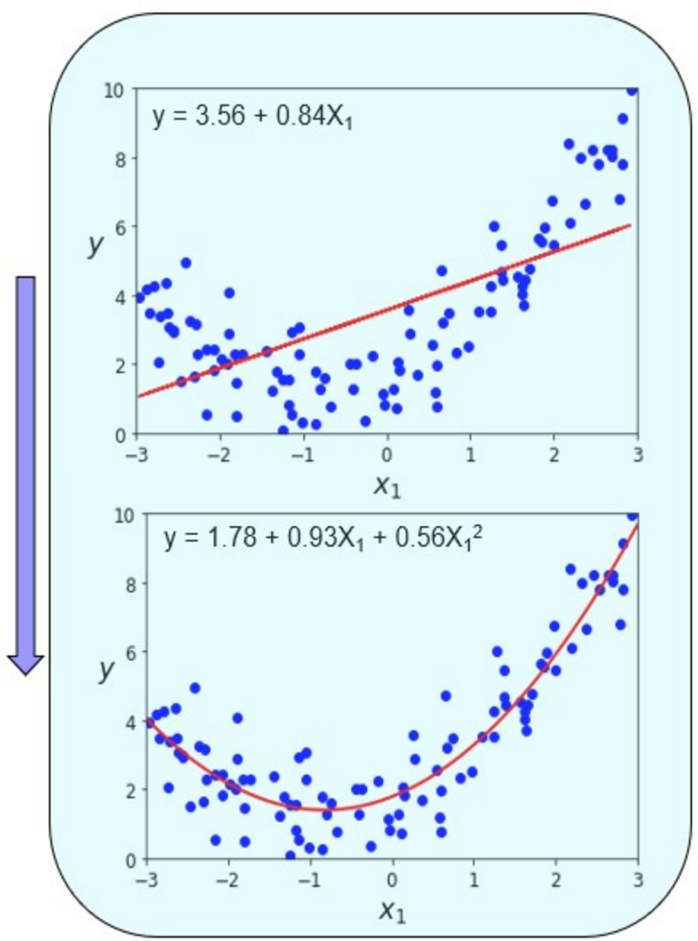


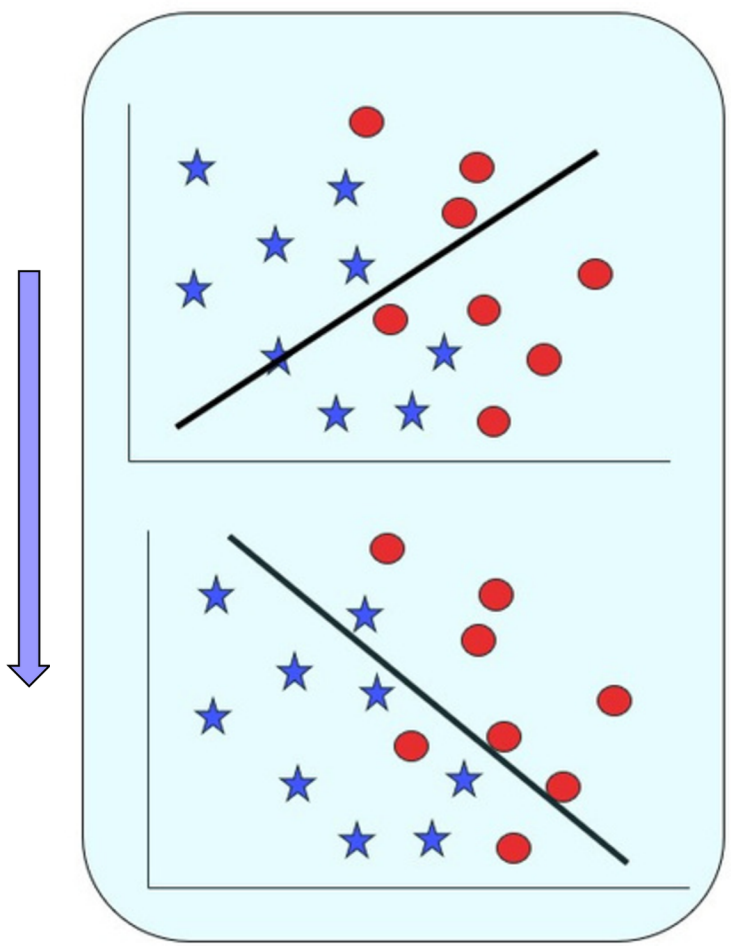


Generalization

The accuracy of the model can be improved by increasing the complexity of the model. E.g., adding a polynomial term, add more features, change model type, remove noise from data, run more training iterations

Simple model, appropriate fit, generalizable →

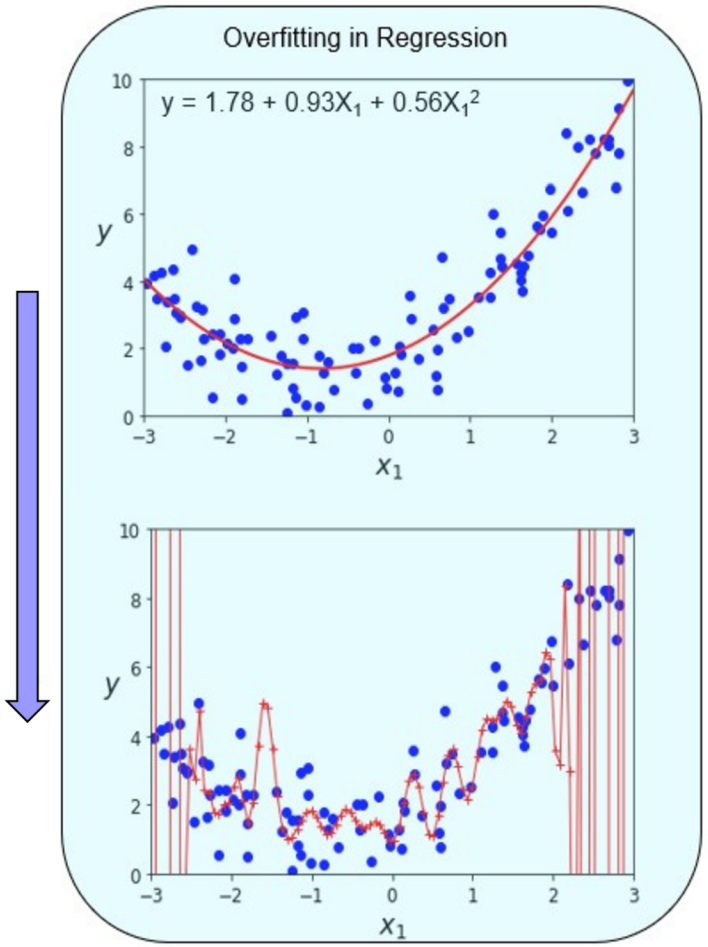


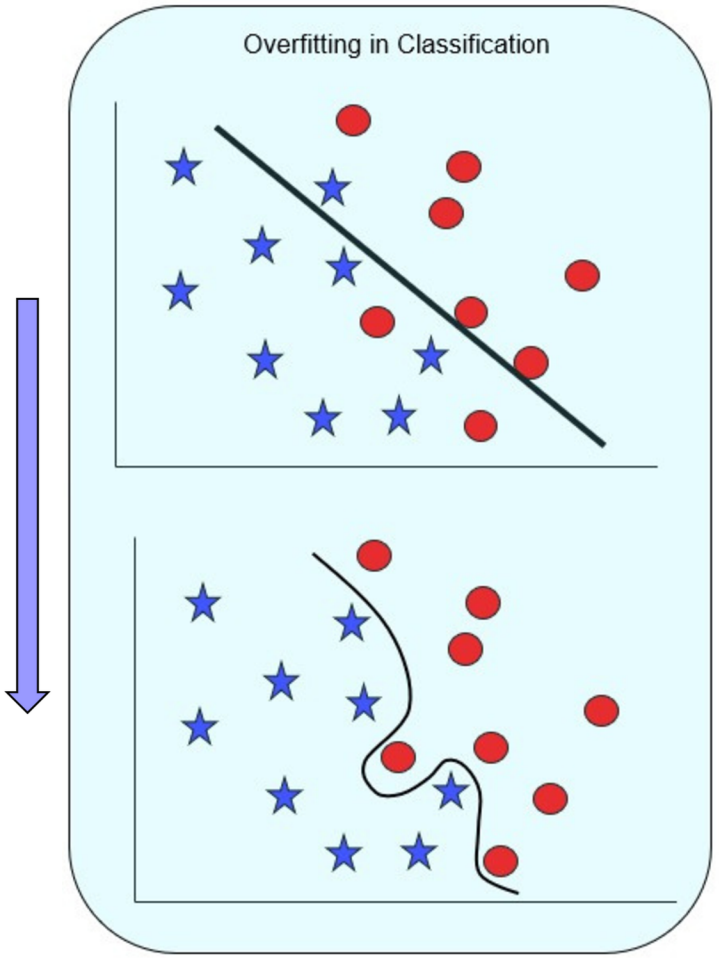


Overfitting

Overfitting happens when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data.

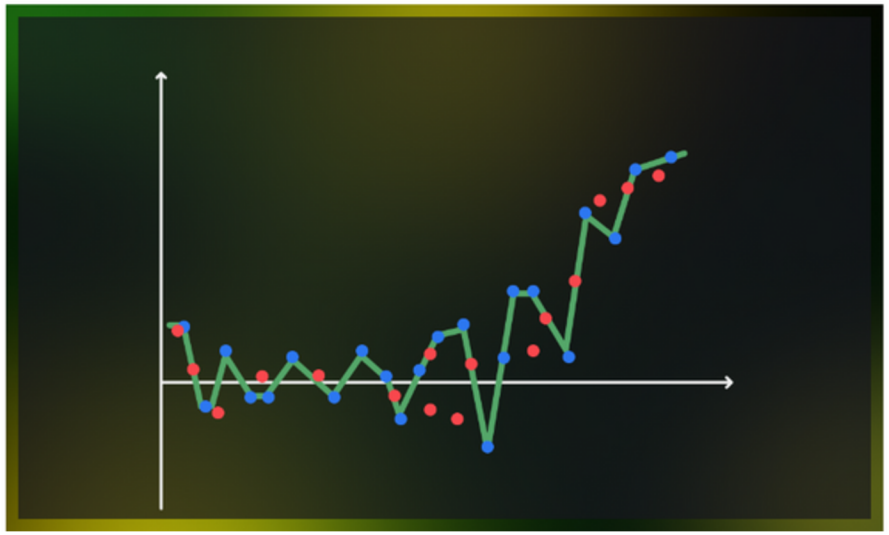
High degree polynomial model, very accurate on training data but not generalizable





How to reduce Variance?

How to reduce high variance?

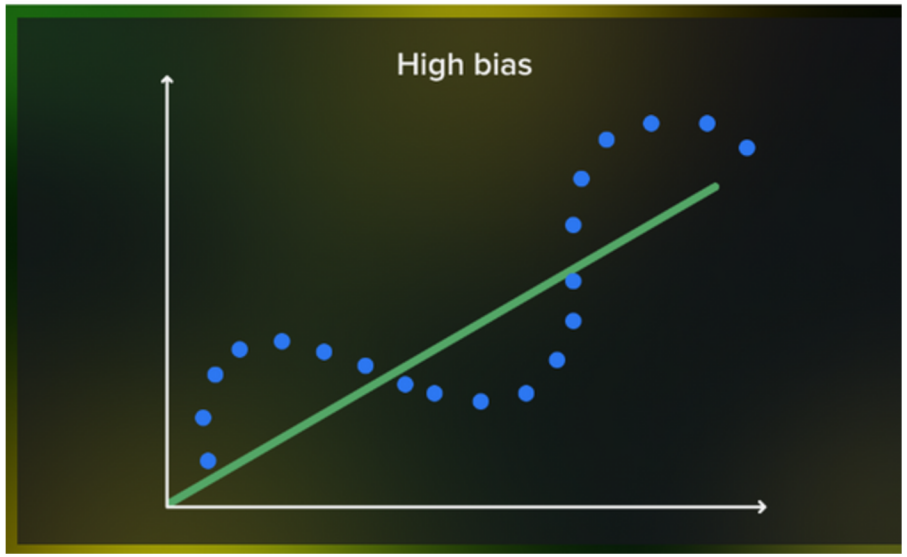


1.Reducing the **number of features** in the model.
2.Replacing the current model with a **simpler one.**
3. Increasing the **training data diversity** to balance out the complexity of the model and the data structure.
4.Performing **hyperparameter tuning** to avoid overfitting.
5.**Increasing regularization** on inputs to decrease the complexity of the model and prevent overfitting.

How do you increase training data diversity?

How to reduce Bias?

How to reduce high bias?



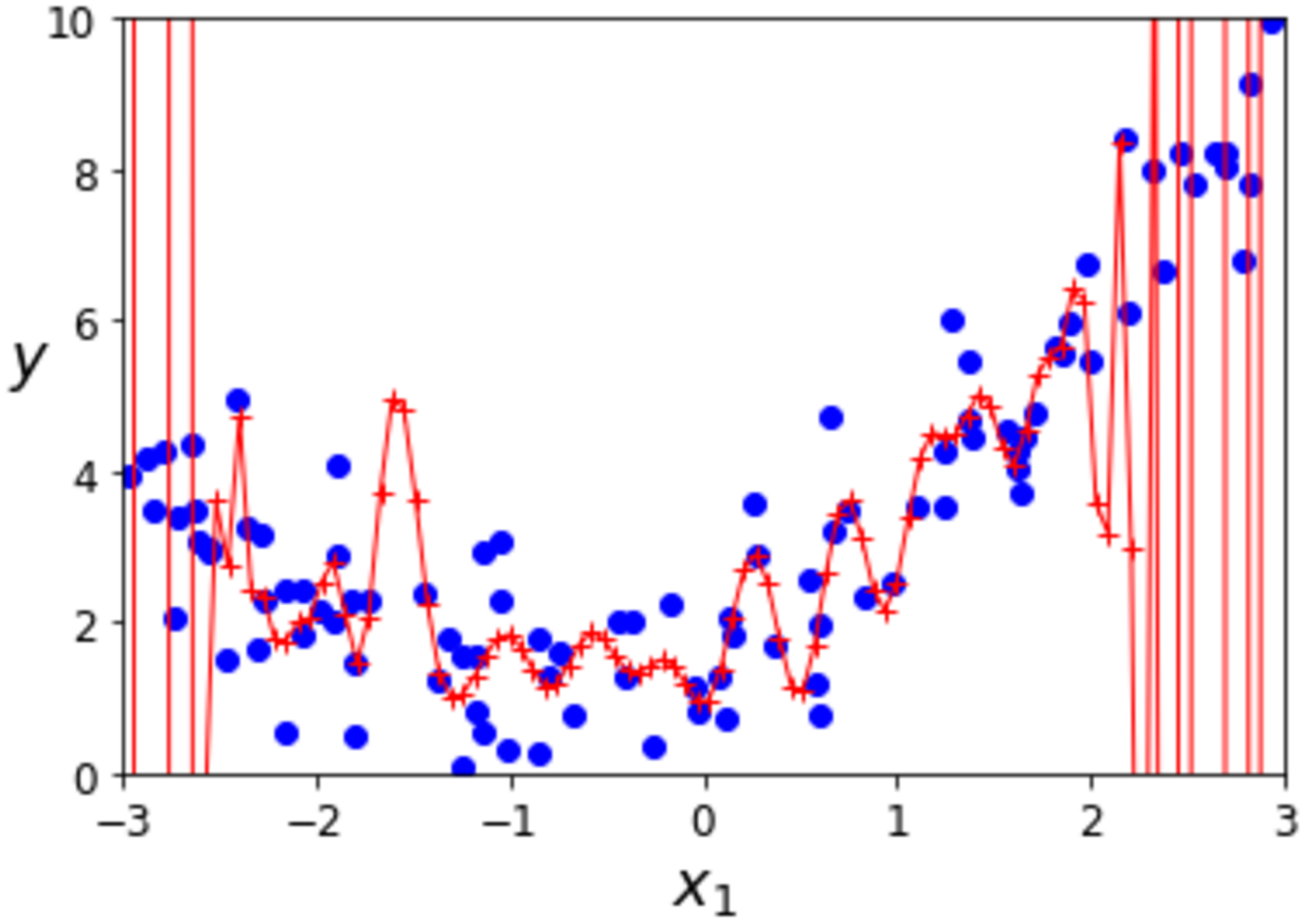
Incorporating additional features from data to improve the model's accuracy.
Increasing the number of training iterations to allow the model to learn more complex data.
Avoiding high-bias algorithms such as linear regression, logistic regression, discriminant analysis, etc. and instead using nonlinear algorithms such as k-nearest neighbors, SVM, decision trees, etc.
Decreasing regularization at various levels to help the model learn the training set more effectively and prevent underfitting.

Regularized Linear Models

Overfitting can be reduced by regularizing models i.e., constrain the model or reduce its degree of freedom

Two methods in linear models:

- Lasso (Least Absolute Shrinkage and Selection Operator) Regression (L1)
- Ridge Regression (L2)



Regularizing models (constrained models)

Ypred = β0+ β1 x1+.....+ βp xp

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2$$

Non-regularized model

During training

$$MSE_{new} = MSE + \alpha(RegTerm)$$

$\alpha$is a hyperparameter

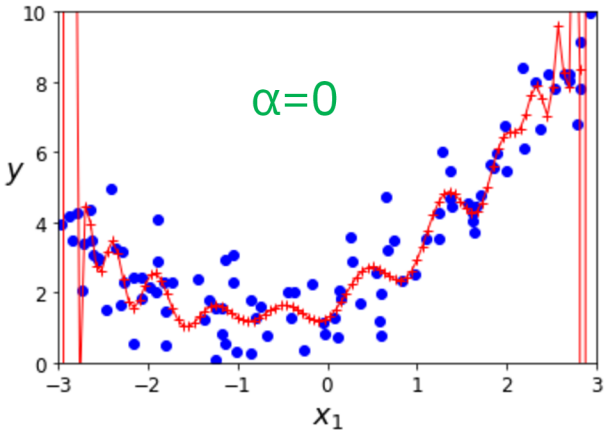
$\sum_{j=1}^{p}|w_j|$

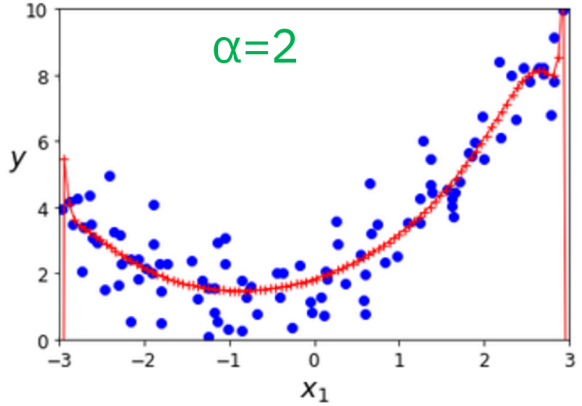
L1 coefficients can be reduced to 0

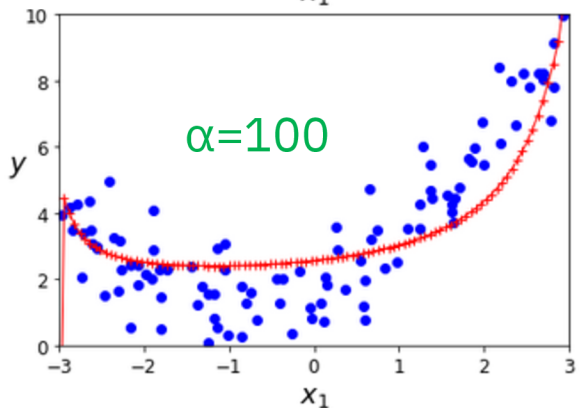
$\sum_{j=1}^{p}w_j^2$ Ridge (L2)

L2 coefficients do not reach 0

Result = The coefficients become smaller (penalized)and the model becomes less complex







Cross Validation an Optimization strategy

3.1. Cross-validation: evaluating estimator performance

Learning the parameters of a prediction function and testing it on the same data is a methodological mistake: a model that would just repeat the labels of the samples that it has just seen would ha...

scikit-learn

What is Cross Validation ?

Cross Validation is a technique of resampling different portions of training data for validation on different iterations.

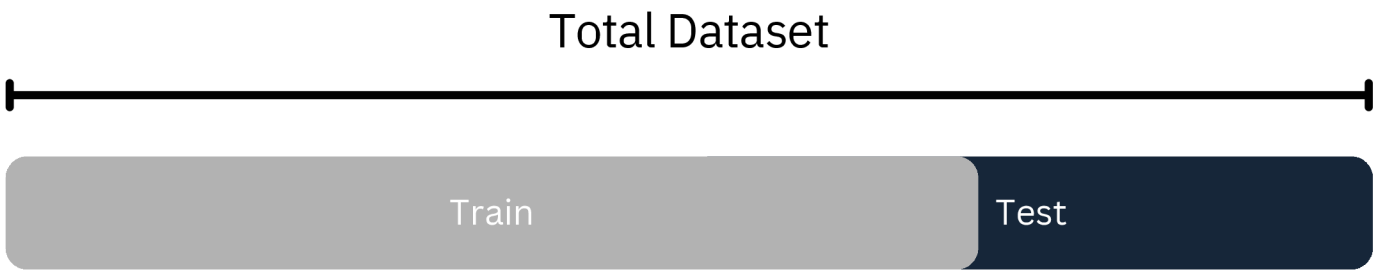
Let’s see the cross-validation methods are

1. Hold-Out
2. K-Folds
3. Stratified K-Folds
4. Repeated k-Fold cross-validation
5. Time-series CV

What is Cross Validation -

Hold -Out Method

```python
from sklearn.model_selection import train_test_split  # Validation , One-hold validation

X = encoded_df.drop(columns=['price'])  # Input features
y = encoded_df['price']                 # Target feature

X_train, X_test , y_train , y_test = train_test_split(X, y , test_size=0.15 , random_state=42)
```

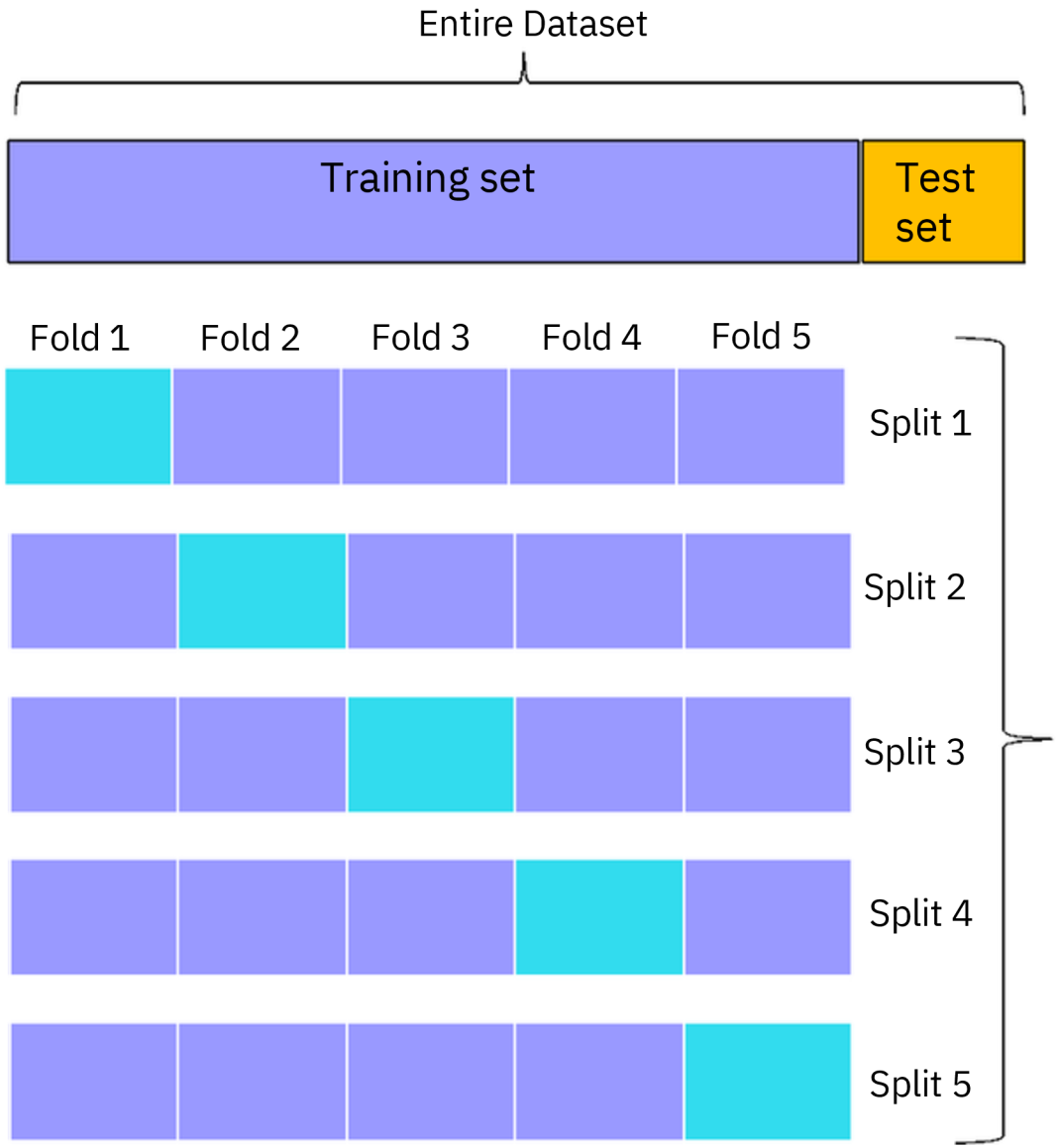


We use this method on large dataset at it requires training the model only once

A dataset with uneven distribution can lead to imbalanced training and test sets after splitting. This may result in poor representation, causing one set to be significantly easier or harder than the other.

What is Cross Validation - KFold

k-Fold cross-validation



Makes better use of all your data and produces more reliable metrics and/or parameters

STEPS:

1.Pick a number offolds –k. Usually, k is 5 or 10.
2.Split the dataset into k equal parts (folds)
3.Choose k –1 folds as the training set. The remaining fold will be the validation set
4.Train the model on the training set
5.Validate on the validation set
6.Save the result of the validation
7.Repeat steps 3 –6 ktimes. In the end, you should have validated the model on every fold that you have.
8.To get the final score average the results that you got on step

9. Confirm results of best model on the Test set

What is Cross Validation - KFold

```python
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeClassifier


train_accuracies        = []
cross_val_accuracies    = []
max_depths              = range(1, 12)


for depth in max_depths:
    model = DecisionTreeClassifier(max_depth=depth, random_state=42)

    cv_scores = cross_val_score(model, X, y, cv=5, scoring='accuracy')   # note we are now pass the full data since we are apply K-FOLD CV
    cross_val_accuracies.append(np.mean(cv_scores))

    model.fit(X, y)
    train_accuracies.append(np.mean(model.predict(X) == y))
```

What is Cross Validation

Stratified K-Fold Cross Validation

Stratified k-Fold is a variation of k-Fold CV designed to handle target imbalance.

It ensures each fold maintains the same class distribution (classification) or similar mean target values (regression) as the full dataset.

- **Analyze Target Distribution** Stratified k-Fold first calculates the class distribution in the dataset. For example, if 60% of the samples belong to Class A and 40% to Class B, this proportion is noted.
- **Allocate Samples Proportionally:** During the splitting process, the method allocates samples to each fold while preserving the overall class proportions.
- For instance, in a 5-fold split: Each fold will contain approximately 60% Class A samples and 40% Class B samples.
- **This prevents any fold from being dominated by one class or missing a class entirely.**

Repeated k-Fold cross-validation

Repeated k-Fold cross-validation (or Repeated random sub-sampling CV) is a robust variation of k-Fold CV. Instead of fixed folds, it trains the model k times using randomly selected test sets.

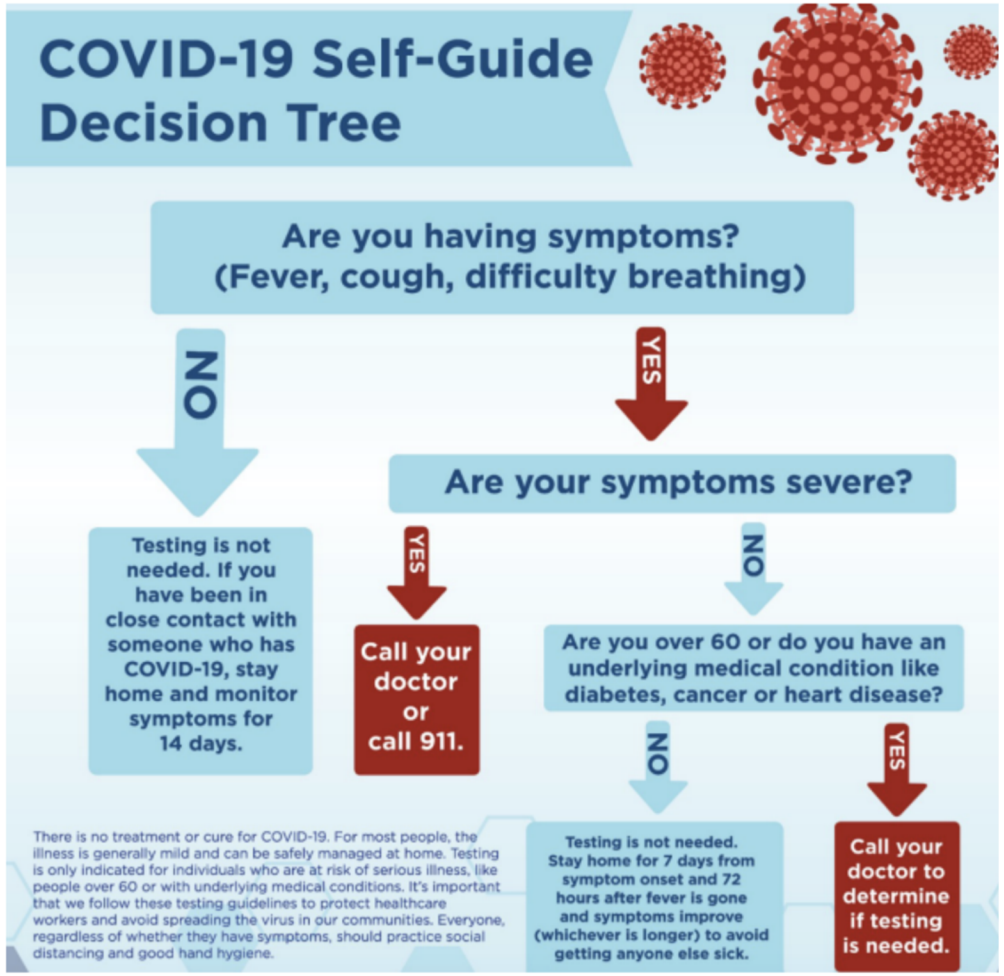
For each iteration:

- Randomly split the dataset into training (e.g., 80%) and test (e.g., 20%) sets.
- Train a new model on the training set.
- Validate on the test set and save the result.
- Repeat K times with different random splits.
- The final score is the average of all validation results.

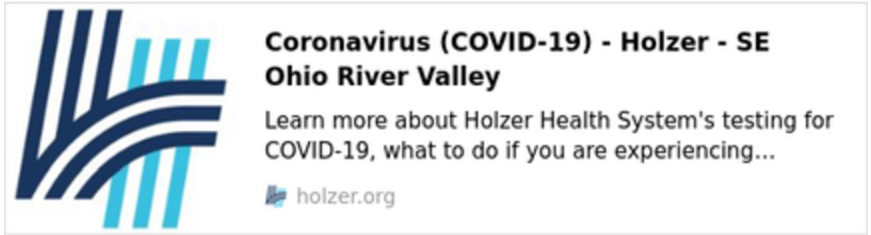
Decision Tree

A decision tree is a series of questions that lead to multiple outcomes such that we can explore each question further.

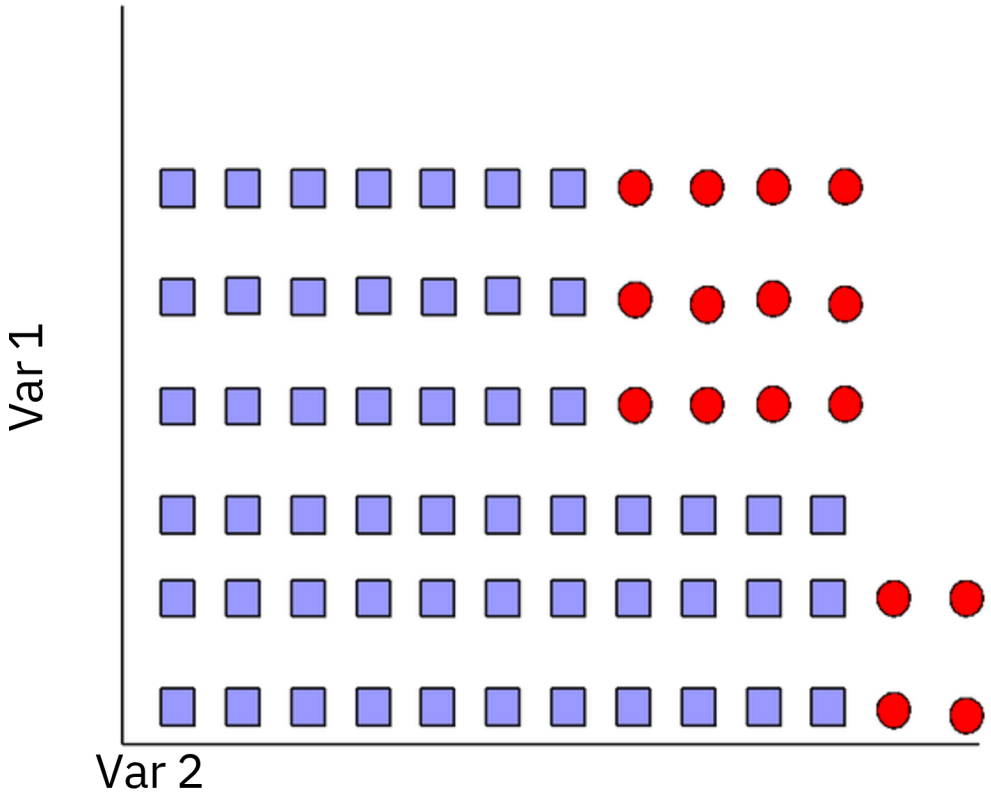
Decision trees are most applicable for nonlinear decision boundaries that occur along a set of distinct criteria



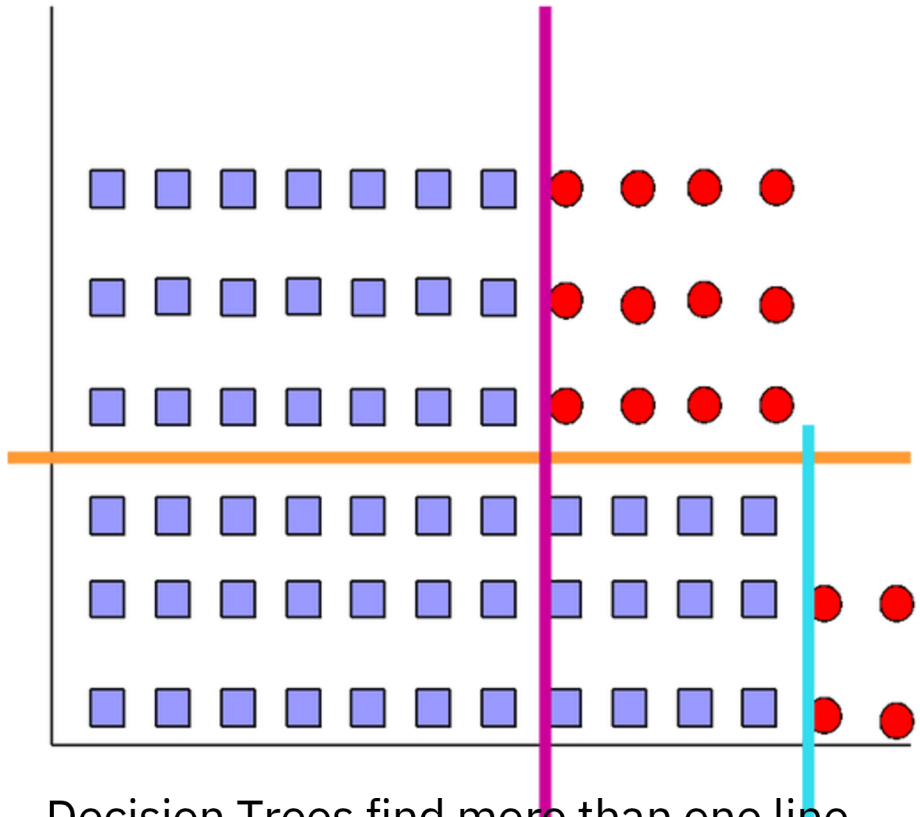
A decision tree on COVID-19 safety guidelines based on a series of questions



Decision Tree



Draw a single line that separates the classes

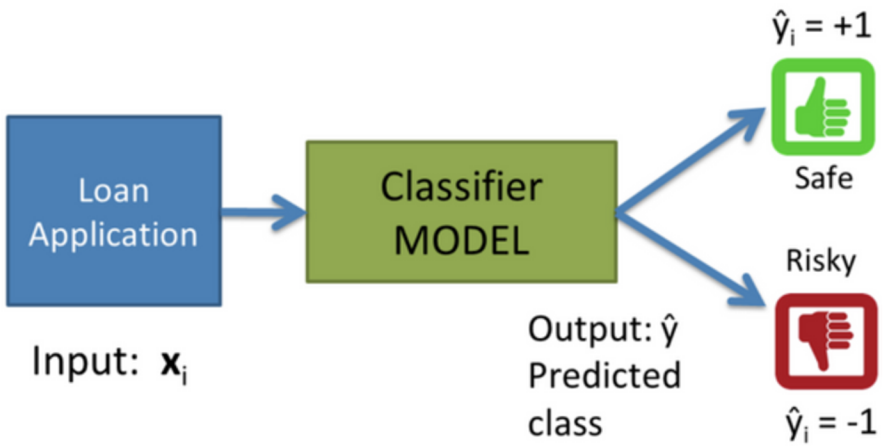


Decision Trees find more than one line to separate the classes

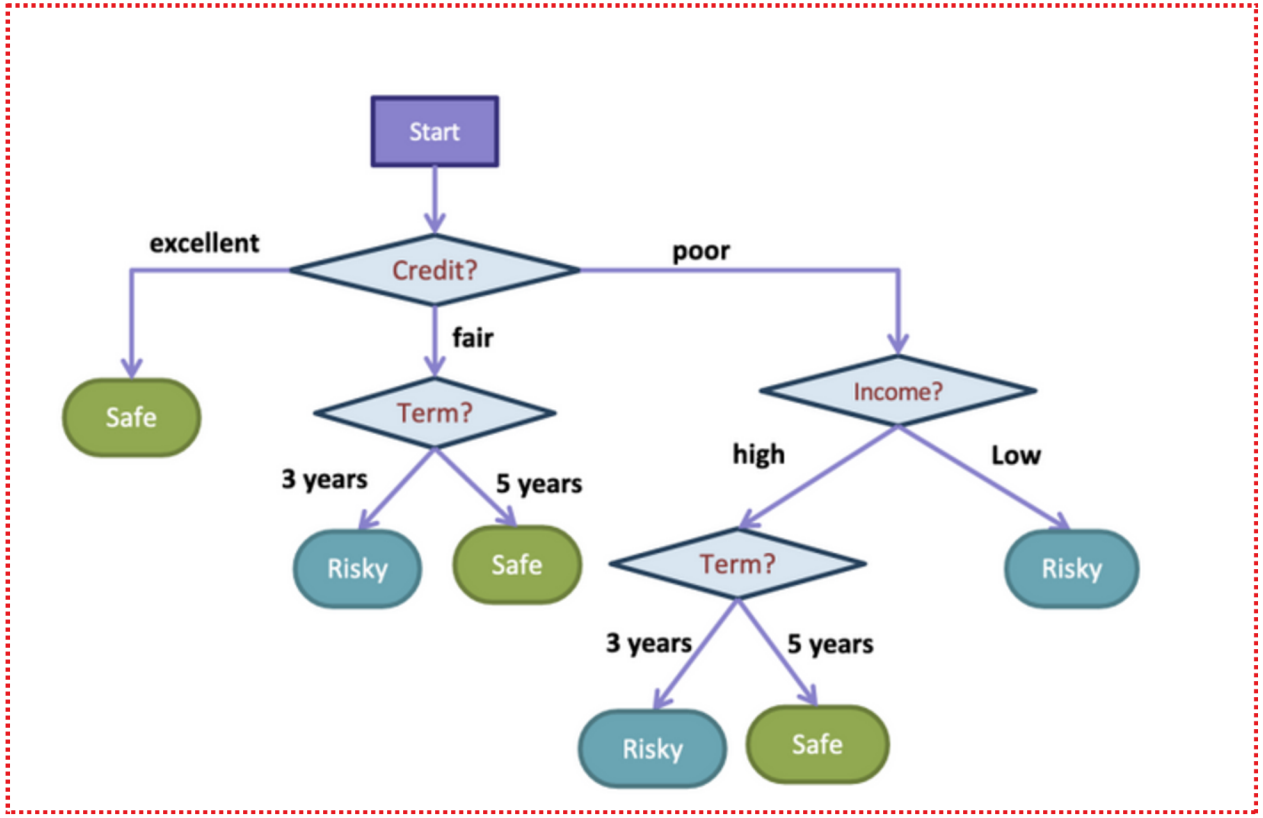
Decision Tree

Credit	Term	Income	y
excellent	3 yrs	high	safe
fair	5 yrs	low	risky
fair	3 yrs	high	safe
poor	5 yrs	high	risky
excellent	3 yrs	low	safe
fair	5 yrs	low	safe
poor	3 yrs	high	risky
poor	5 yrs	low	safe
fair	3 yrs	high	safe

: Let our training dataset be n number of inputs and output pairs. Each input has 3 features: credit, term and income and one output, y. After training on the train dataset, we aim to reach an accurate prediction ˆy of whether a new candidate is safe or risky based on their loan application: A decision tree for this example can be set up like so:

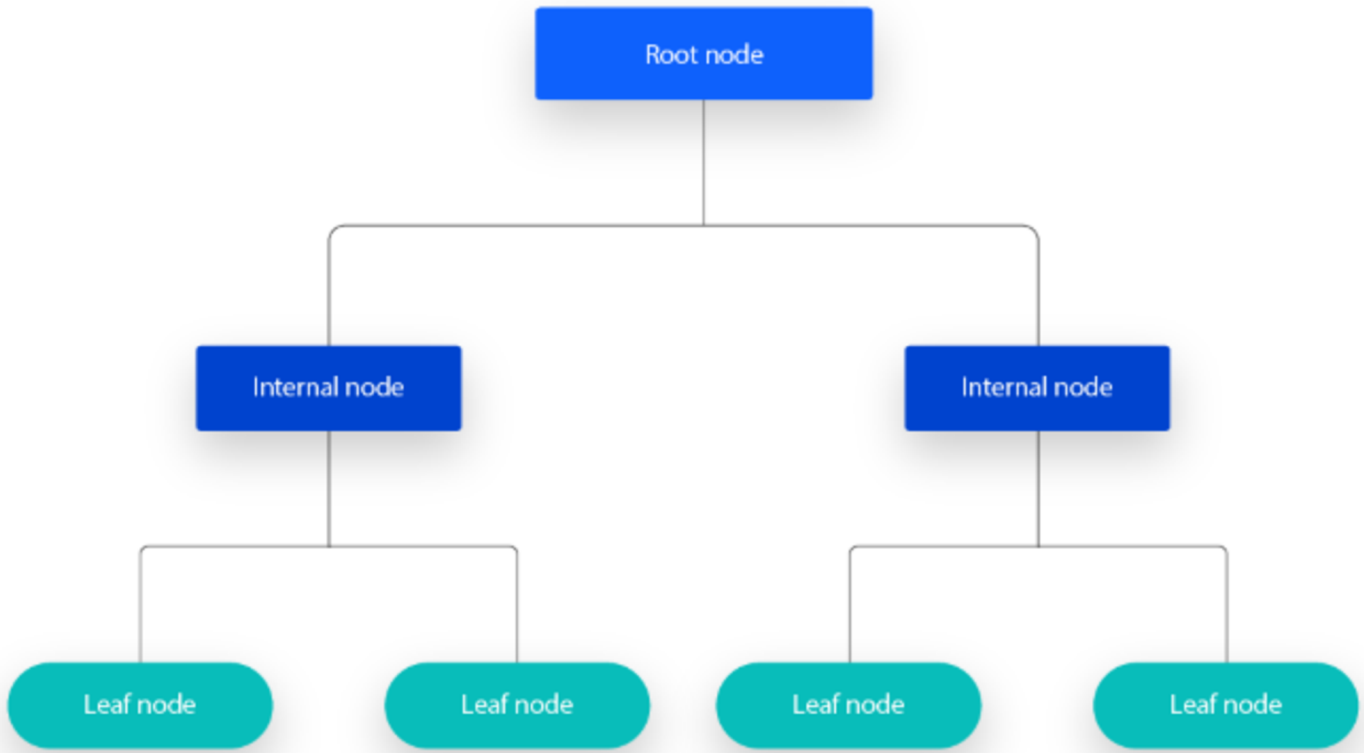


Decision Tree



the model first splits up loan applications based on credit history. If the credit history is excellent, a candidate's application is considered safe. However, if their credit history is poor, the tree splits into a more complex structure as more questions are required to determine if the candidate should receive a loan

Decision Tree



What is a Decision Tree?

A decision tree is a non-parametric supervised learning algorithm, which is utilized for both classification and regression tasks.

ibm.com / Dec 19, 2024

Decision Tree - Take Example

Feature Selection Method

ID	Age	Income	Credit	Buys_computer
1	Youth	Low	Good	No
2	Senior	High	Bad	No
3	Senior	High	Good	Yes
4	Youth	Low	Good	Yes
5	Youth	High	Good	Yes
6	Senior	High	Bad	No
7	Senior	High	Bad	No
8	Senior	Low	Bad	No
9	Youth	Low	Bad	Yes
10	Senior	High	Good	Yes

A. Calculate Entropy for root node and for each feature

$H(X) = -[(pi* \log2pi) + (qi* \log2qi)]$

1. Entropy @ Training set

$H(D) = -((5/10)\log2(5/10) + (5/10)\log2(5/10)) =$ 1.00 bits

2. $H(Age) = (4/10) * (-((3/4)\log2(3/4) + (1/4)\log2(1/4)))$
$+ (6/10) * (-((2/6)\log2(2/6) + (4/6)\log2(4/6))) =$ 0.8755

3. $H(Income) = (4/10) * (-((2/4)\log2(2/4) + (2/4)\log2(2/4)))$
$+ (6/10) * (-((3/6)\log2(3/6) + (3/6)\log2(3/6))) =$ 1.0000

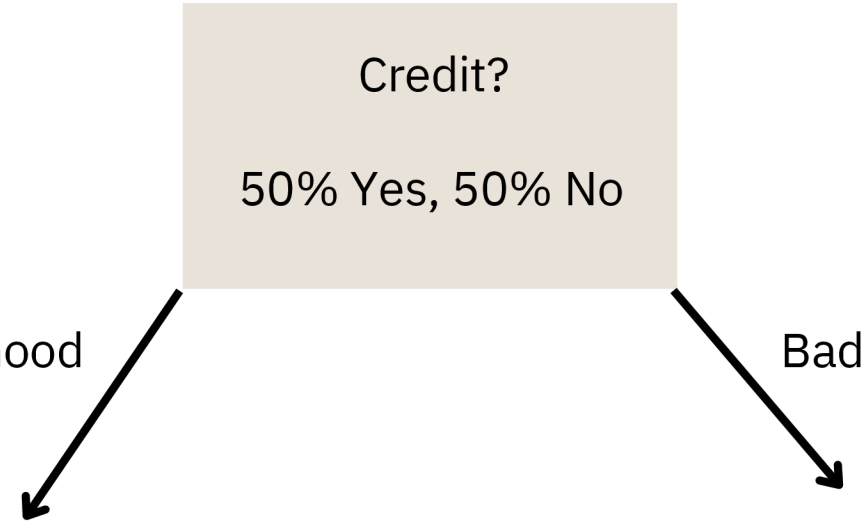
4. $H(Credit) = (5/10) * (-((4/5)\log(4/5) + (1/5)\log(1/5)))$
$+ (5/10) * (-((1/5)\log2(1/5) + (4/5)\log2(4/5))) =$ 0.7219

B. Calculate Information Gain and compare

IG (Age) = 1-0.8755 = 0.1245
IG(Income) = 1-1 = 0.0000
IG(Credit) = 1-0.7219 =0.2781

Highest Information Gain!!!

Decision Tree



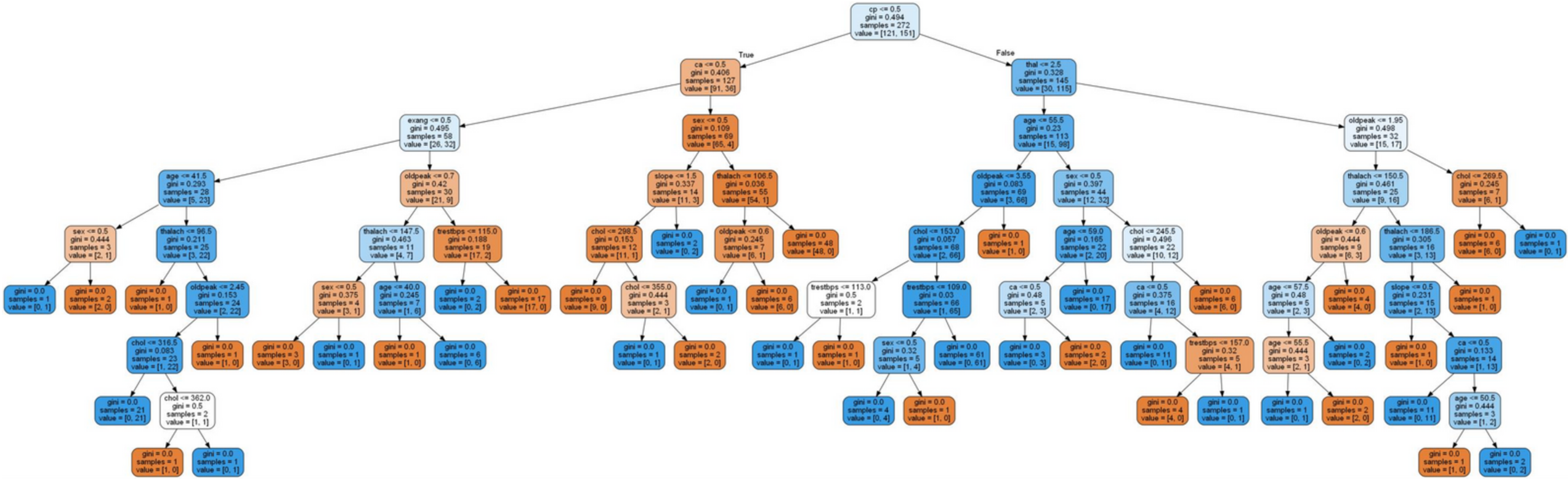
ID	Age	Income	Credit	Buys_computer
1	Youth	Low	Good	No
3	Senior	High	Good	Yes
4	Youth	Low	Good	Yes
5	Youth	High	Good	Yes
10	Senior	High	Good	Yes

80% YES

ID	Age	Income	Credit	Buys_computer
2	Senior	High	Bad	No
6	Senior	High	Bad	No
7	Senior	High	Bad	No
8	Senior	Low	Bad	No
9	Youth	Low	Bad	Yes

20% YES

Decision Tree



Train accuracy: 1.0 Test accuracy: 0.71

Overfitting in Decision Trees

Decision Tree

How to Deal with Overfitting ???

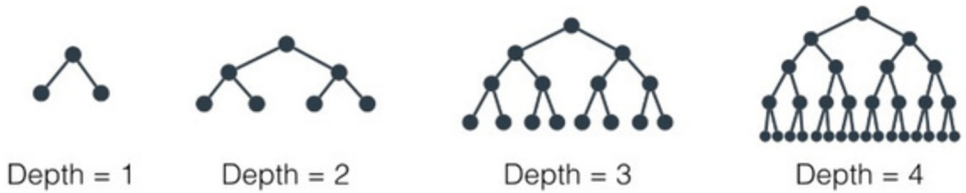
- Regularization
  - Pruning
    - Pre-Pruning (early stopping)
    - Post-Pruning
  - Ensembling

Pre-Pruning Techniques

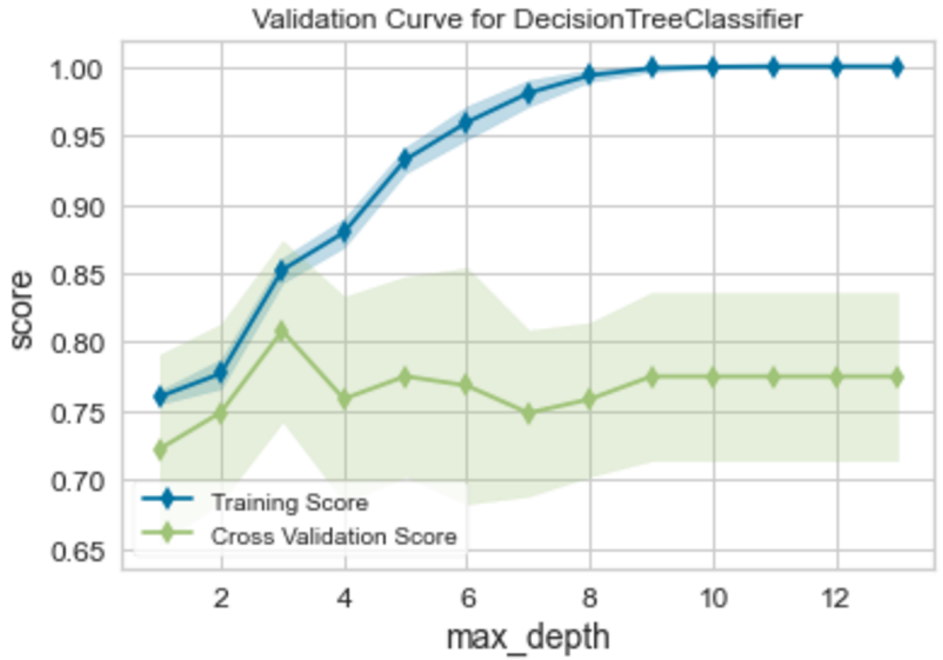
1. Maximum Depth Of The Tree (max_depth):

•Specifies the maximum depth of the tree.

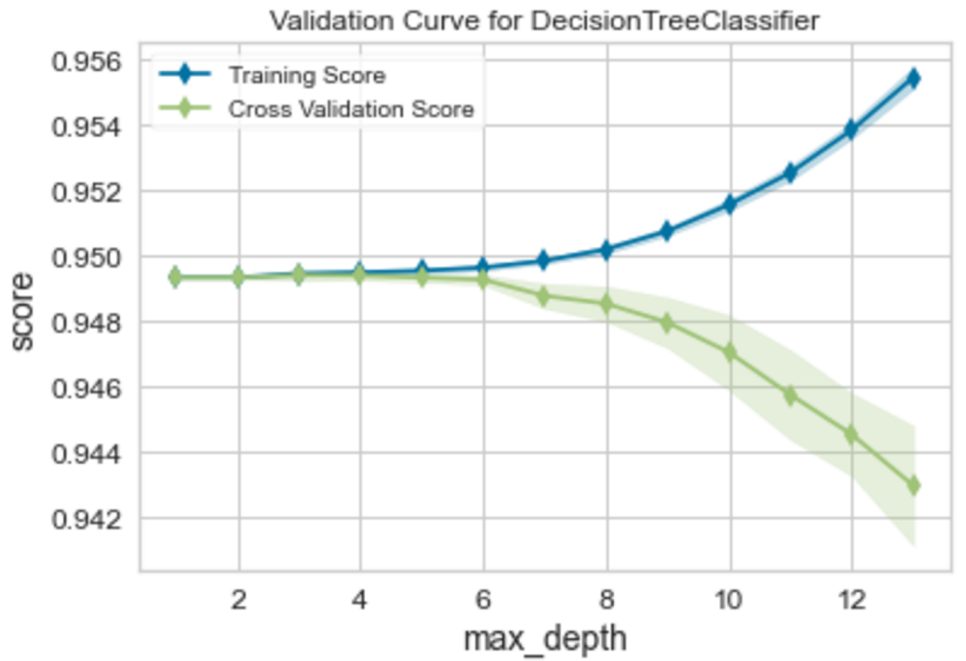
•Controls the complexity of branching (i.e., the number of times the splits are made). Decreasing this value prevents overfitting.



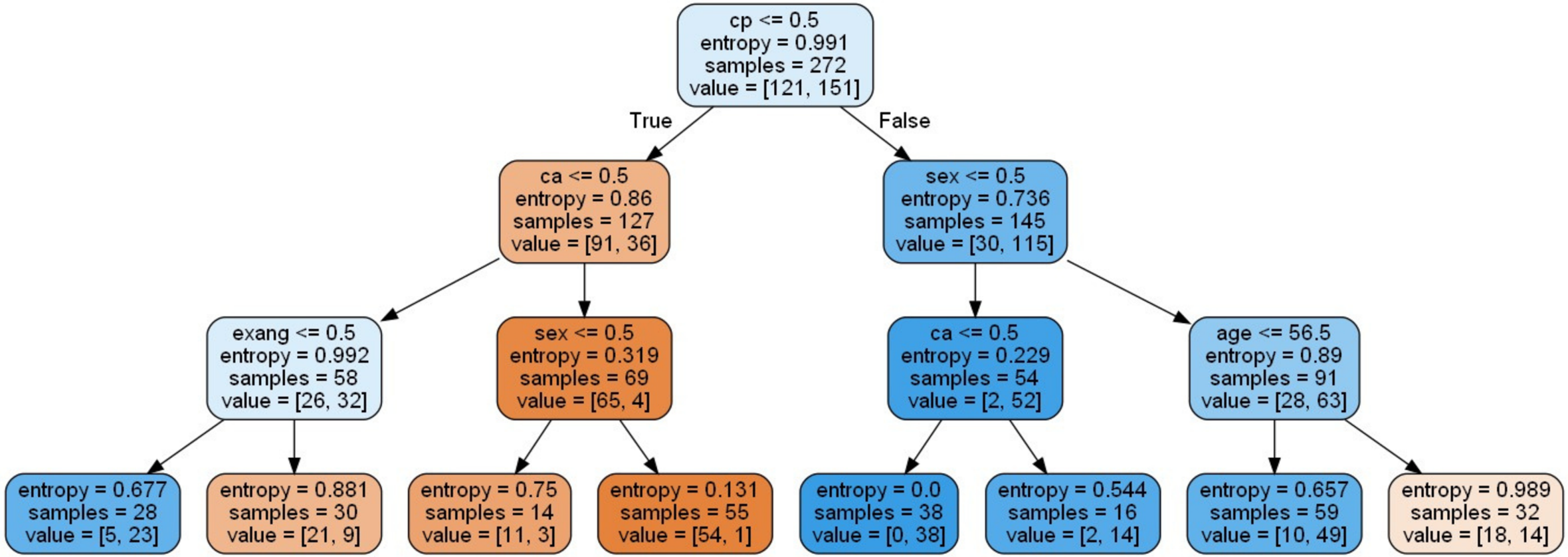
Maximum depth of a decision tree



Look at the charts showing data from 2 different models... what max_depthvalue would you use for each of these models



Decision Tree



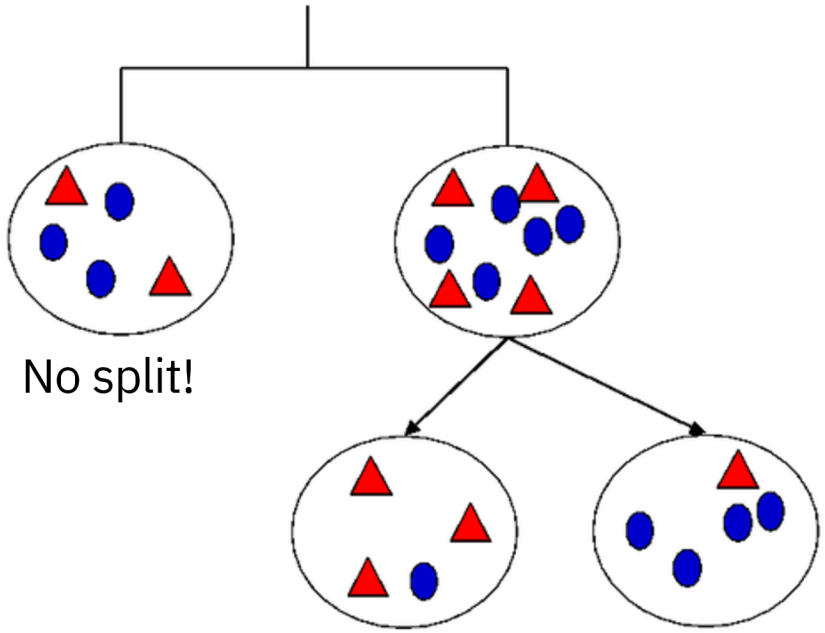
Train accuracy: 0.84 Test accuracy: 0.74

Other Pre-Pruning Techniques

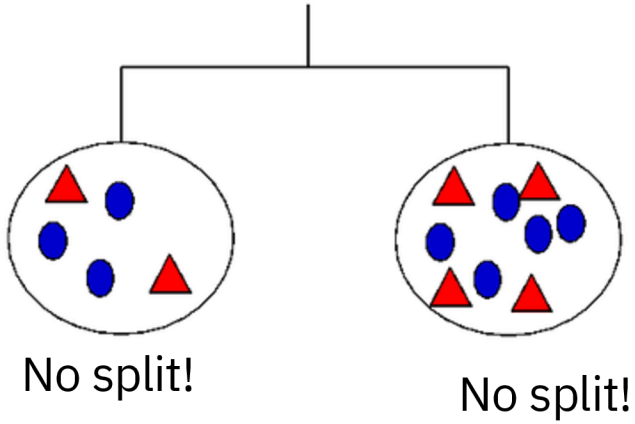
min_samples_split : Specifies the minimum number of samples required to split an internal node. The default is 2

min_samples_leaf Determines the minimum number of samples required to be at a leaf node. The default is 1.

:

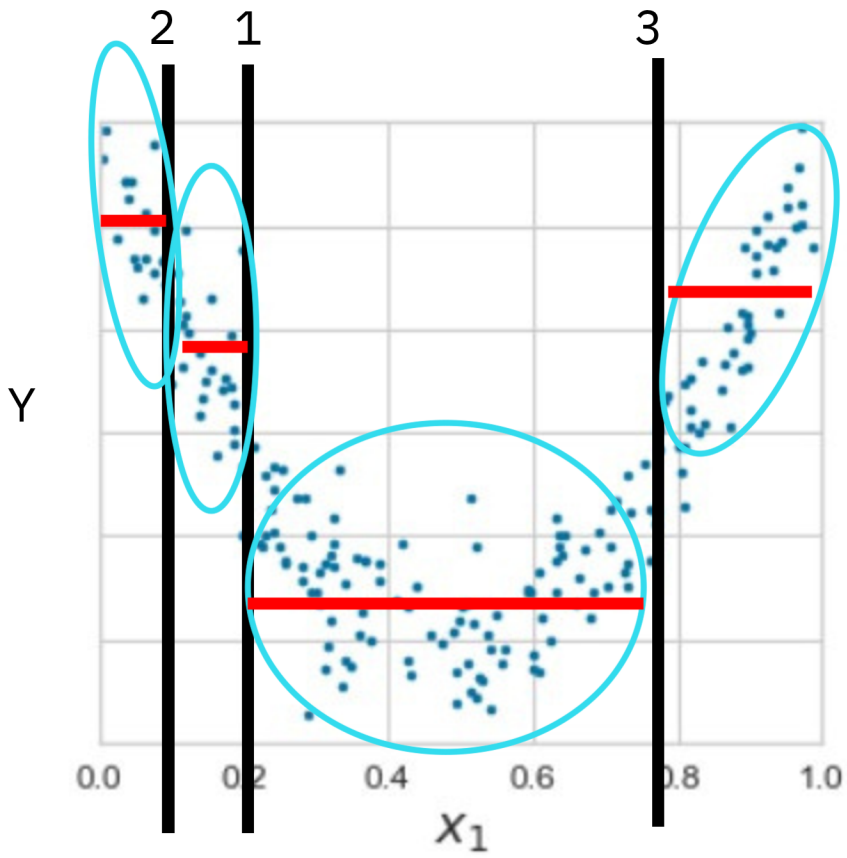


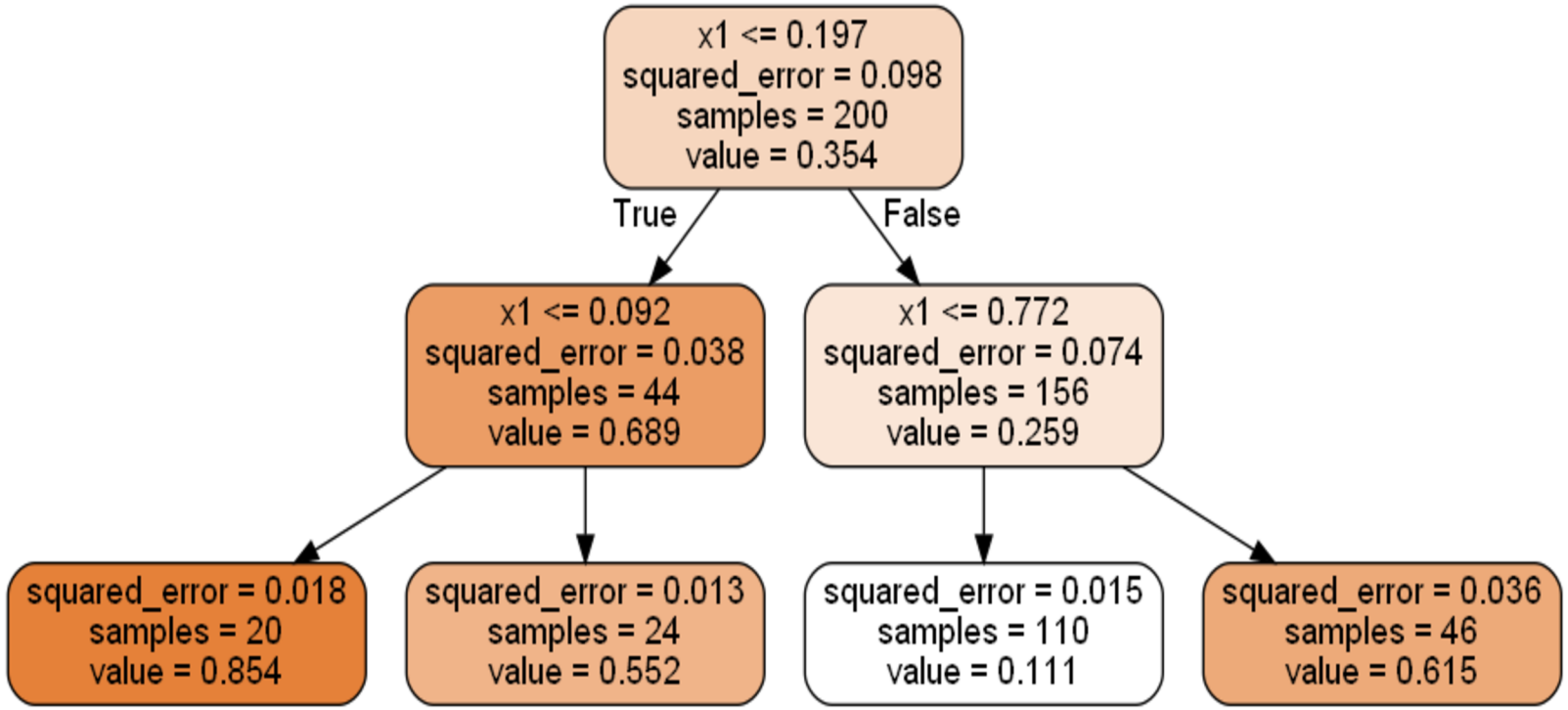
If Min_samples_split= 6



If Min_samples_leaf= 6

DECISION TREE REGRESSION

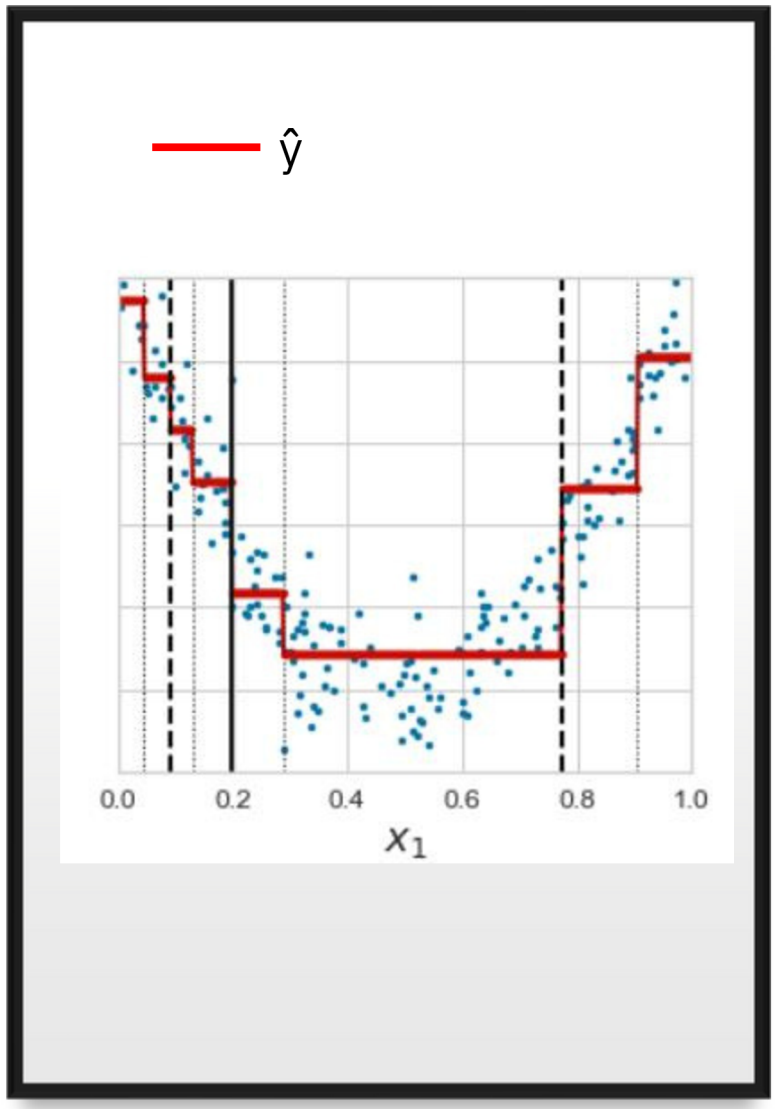


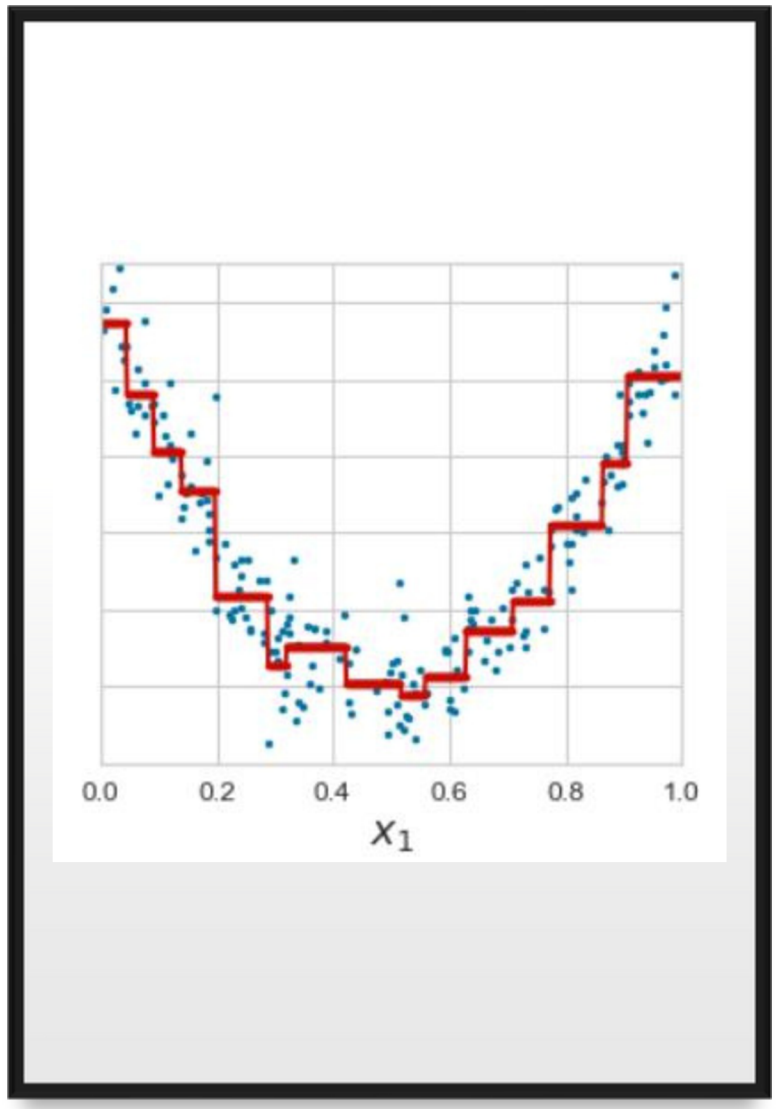


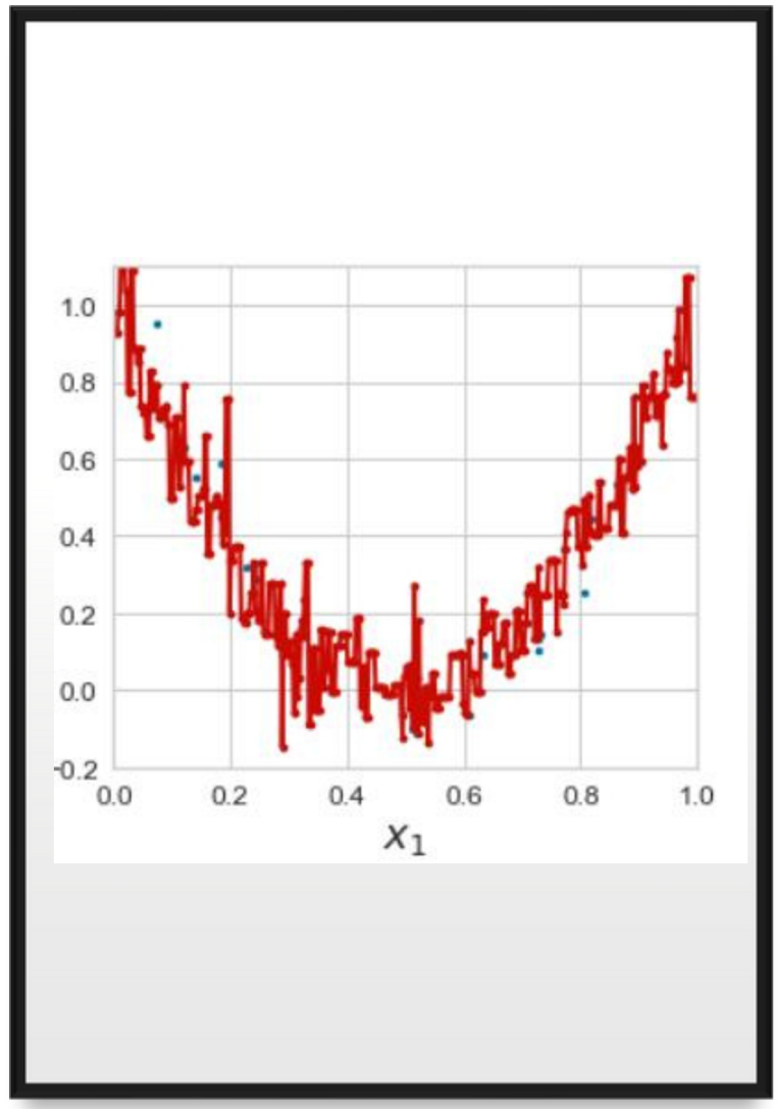
Predictions are made as the average of Y for all samples in the node
Cuts are made to produce the lowest MSE (squared deviations of predicted y from the actual Y)

Decision Tree Regressor

Overfitting ⟶







Ensemble Learning in ML

Ensemble Learning in ML

Motivation

No Free Lunch Theorem

Some models will work better in some scenarios and poorly in other scenarios

Wisdom of the Crowd

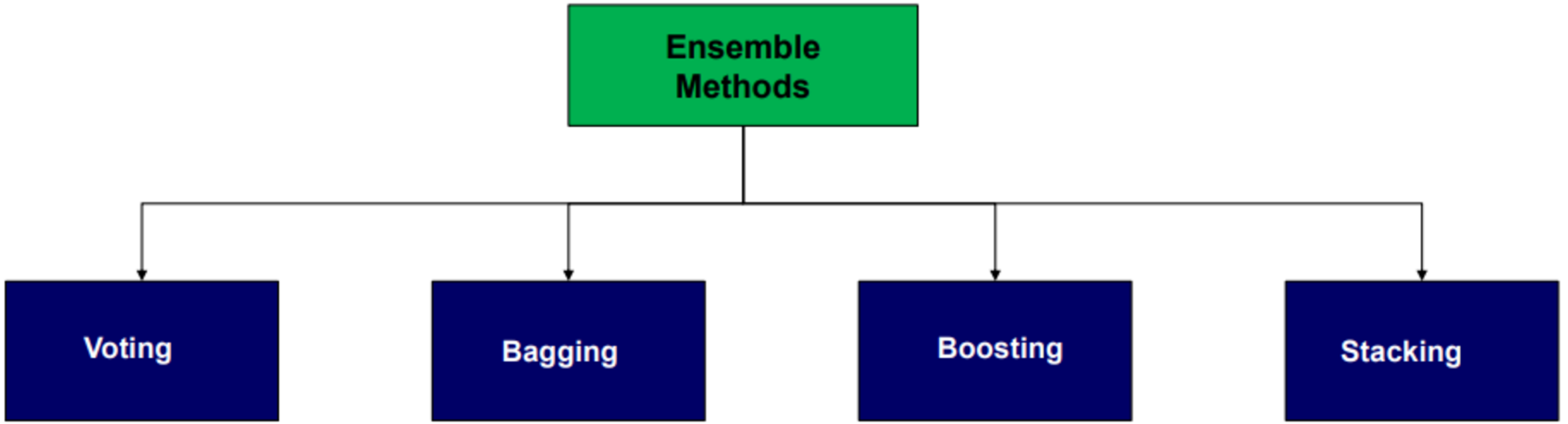
If you aggregate decisions from several predictors, you will get better predictions than with the best individual predictor

Ensemble Learning in ML

- **Ensemble methods** combine several base algorithms to construct better predictive performance than a single tree base algorithm

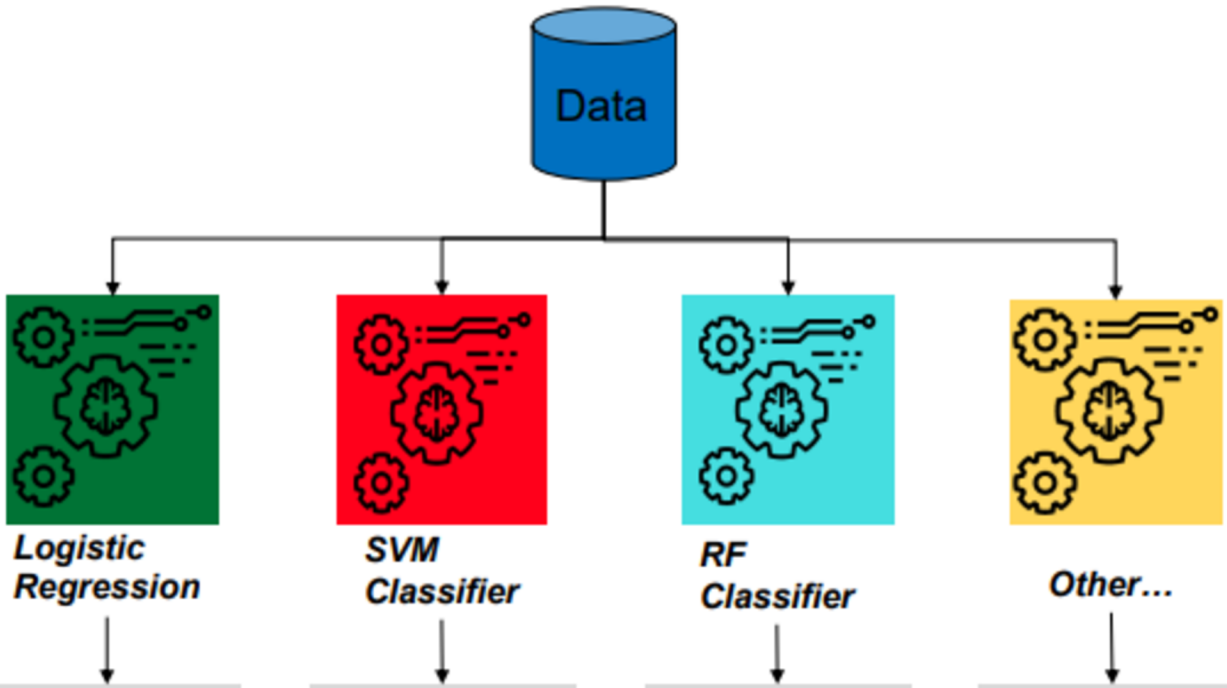
- **Reduce bias:**
- **Reduce variance:** Ensemble methods average predictions from multiple models, reducing variance and leading to more robust predictions.
- **Improve generalizability:** By combining diverse models, ensemble methods can capture a wider range of patterns and make more accurate predictions on unseen data.

Ensemble Learning in ML



Voting

Cat or Not Cat?



Predicted Probabilities	**(0.8, 0.2)**	**(0.4, 0.6)**	**(0.7, 0.3)**	**(0.6, 0.4)**	$(P_r(\text{Yes}), P_r(\text{No}))$
Hard Voting Prediction	**YES**	**NO**	**YES**	**YES**	3 votes 'Yes', 1 vote 'No' Winner = YES
Soft Voting Prediction ***e.g., weight w_i= 0.25***	0.25*0.8 0.25*0.2	0.25*0.4 0.25*0.6	0.25*0.7 0.25*0.3	0.25*0.6 0.25*0.4	Sum of weighted votes 0.62 ✓ **YES** 0.38

Activate W
Go to Settin

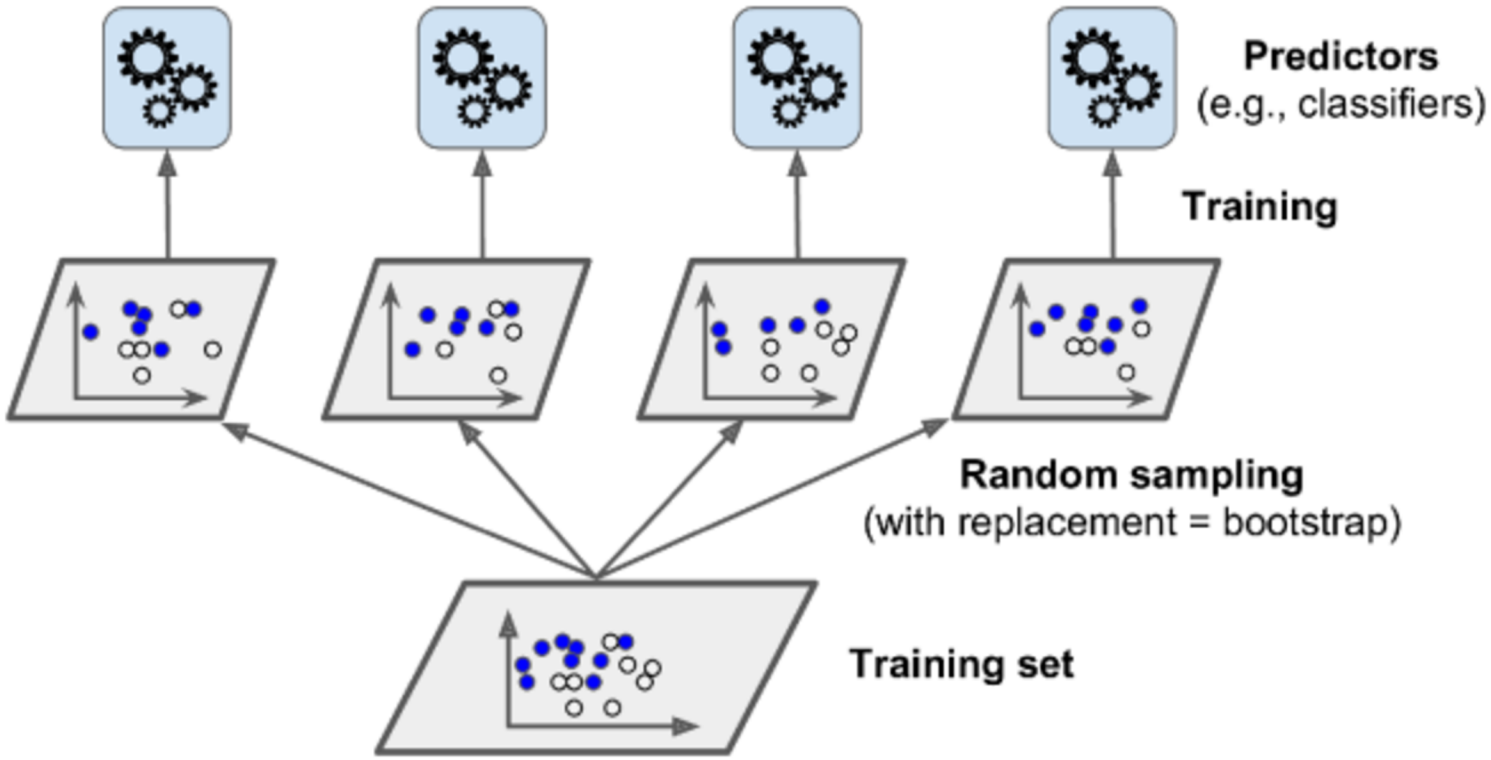
Voting <CODE/>

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC

log_clf = LogisticRegression()
rnd_clf = RandomForestClassifier()
svm_clf = SVC()

voting_clf = VotingClassifier(
    estimators=[('lr', log_clf), ('rf', rnd_clf), ('svc', svm_clf)],
    voting='hard')
voting_clf.fit(X_train, y_train)
```

Evaluate

```
>>> from sklearn.metrics import accuracy_score
>>> for clf in (log_clf, rnd_clf, svm_clf, voting_clf):
...     clf.fit(X_train, y_train)
...     y_pred = clf.predict(X_test)
...     print(clf.__class__.__name__, accuracy_score(y_test, y_pred))
```

Bagging and Pasting



sampling is performed with replacement, this method is called bagging.

sampling is performed withOUT replacement, this method is called pasting

NB. Reduce overfitting by averaging out individual predictions(Variance reduction)

Deep trees: keep low bais , reduce variance

Bagging & Pasting

<CODE/>

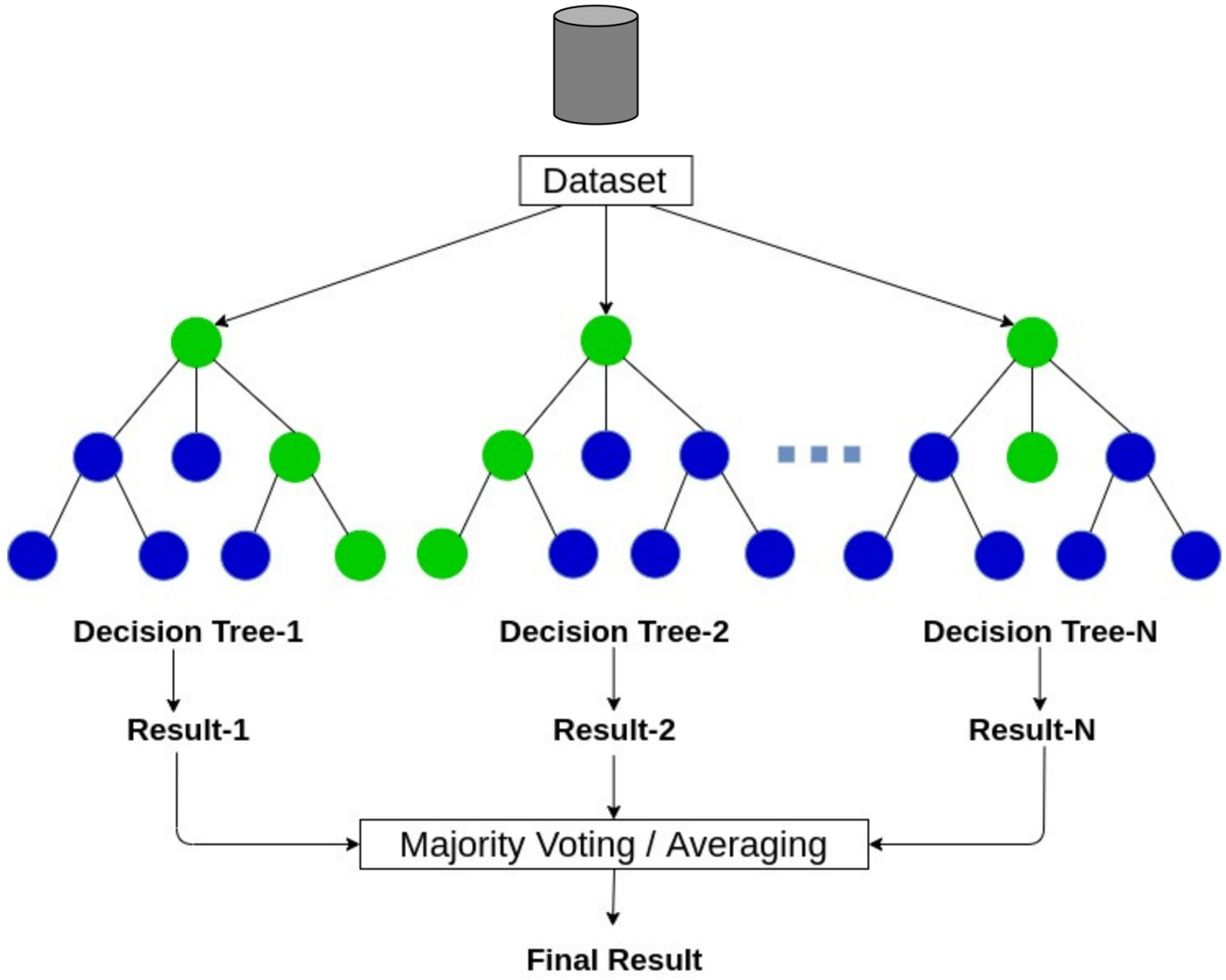
```
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier

bag_clf = BaggingClassifier(
    DecisionTreeClassifier(), n_estimators=500,
    max_samples=100, bootstrap=True, n_jobs=-1)
bag_clf.fit(X_train, y_train)
y_pred = bag_clf.predict(X_test)
```

Random Forest

Trees (DT) having been trained on bootstrap random subsets of the data.

The bootstrap random sampling is applied to both rows (records) and **columns (features)**

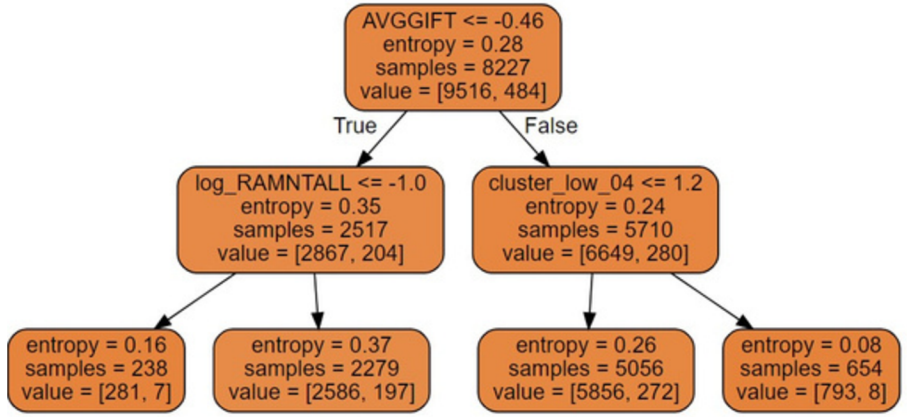
RF are optimized for Decision trees and can be trained in parallel on different cores, predictions can also be made in parallel



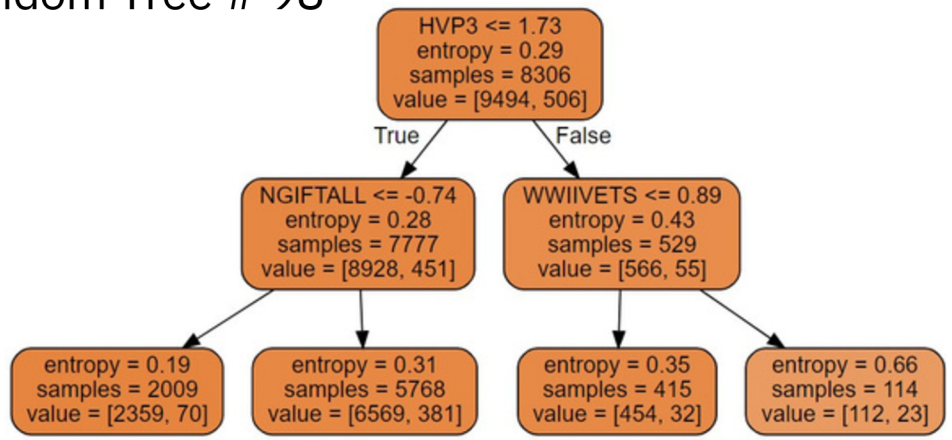
Random Forest

Example of 2 random trees built within an RF of 100 trees

Random Tree # 8



Random Tree # 98



Feature importance within the tree

	importance
AVGGIFT	0.419029
cluster_low_04	0.417630
log_RAMNTALL	0.163341

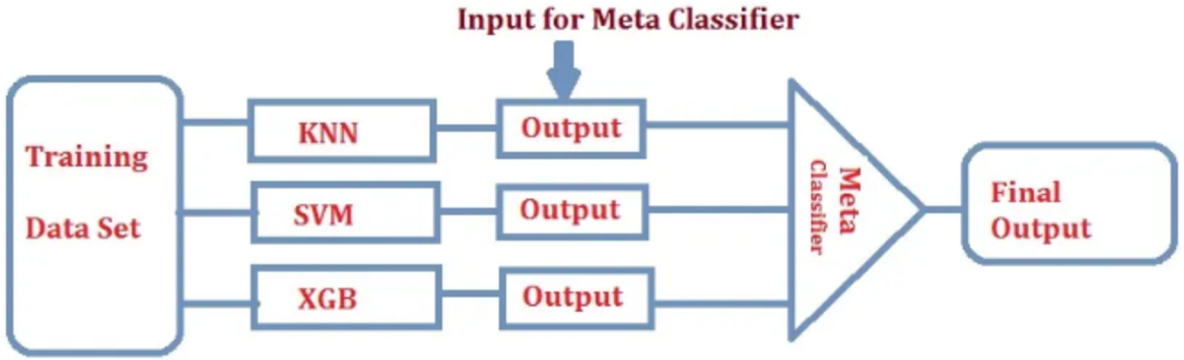
	importance
NGIFTALL	0.503573
HVP3	0.284258
WWIIVETS	0.212169

Stacking Ensemble

Stacking

- Combines Multiple Classifications or regression doels via meta- classifier

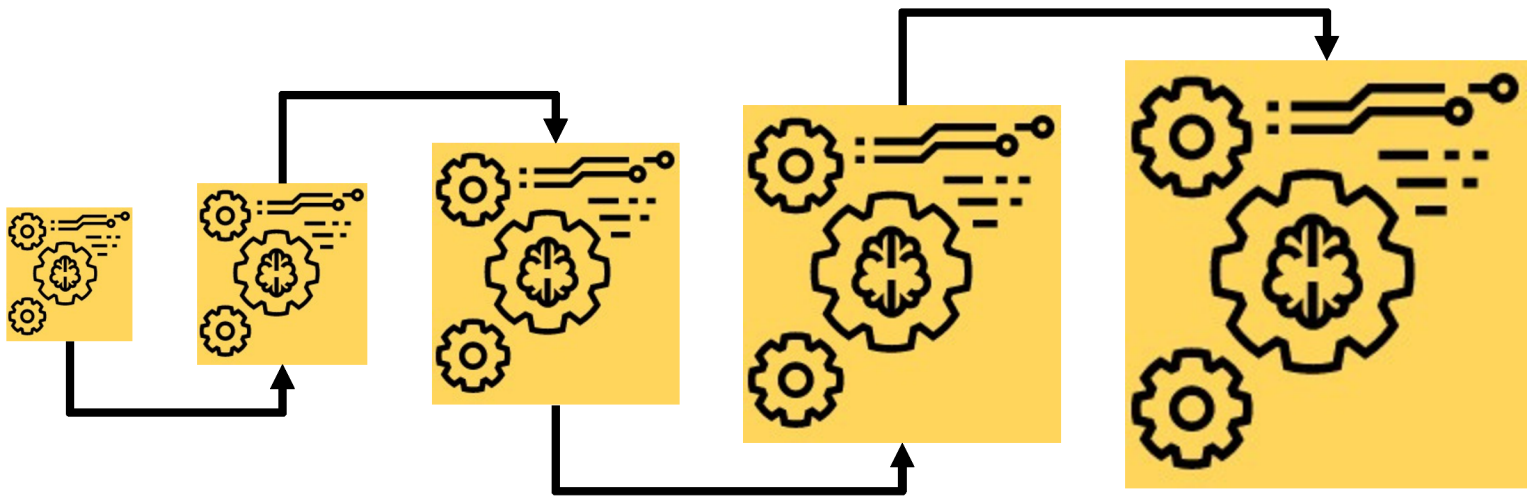
- Combines Multiple Classifications or regression mdoels via meta- classifier



NB. Reduce overfitting by averaging out individual predictions(Variance reduction)
Deep trees: keep low bais , reduce variance

Boosting Ensemble

- Ensemble methods which can sequentially combine several weak learners into a strong learner

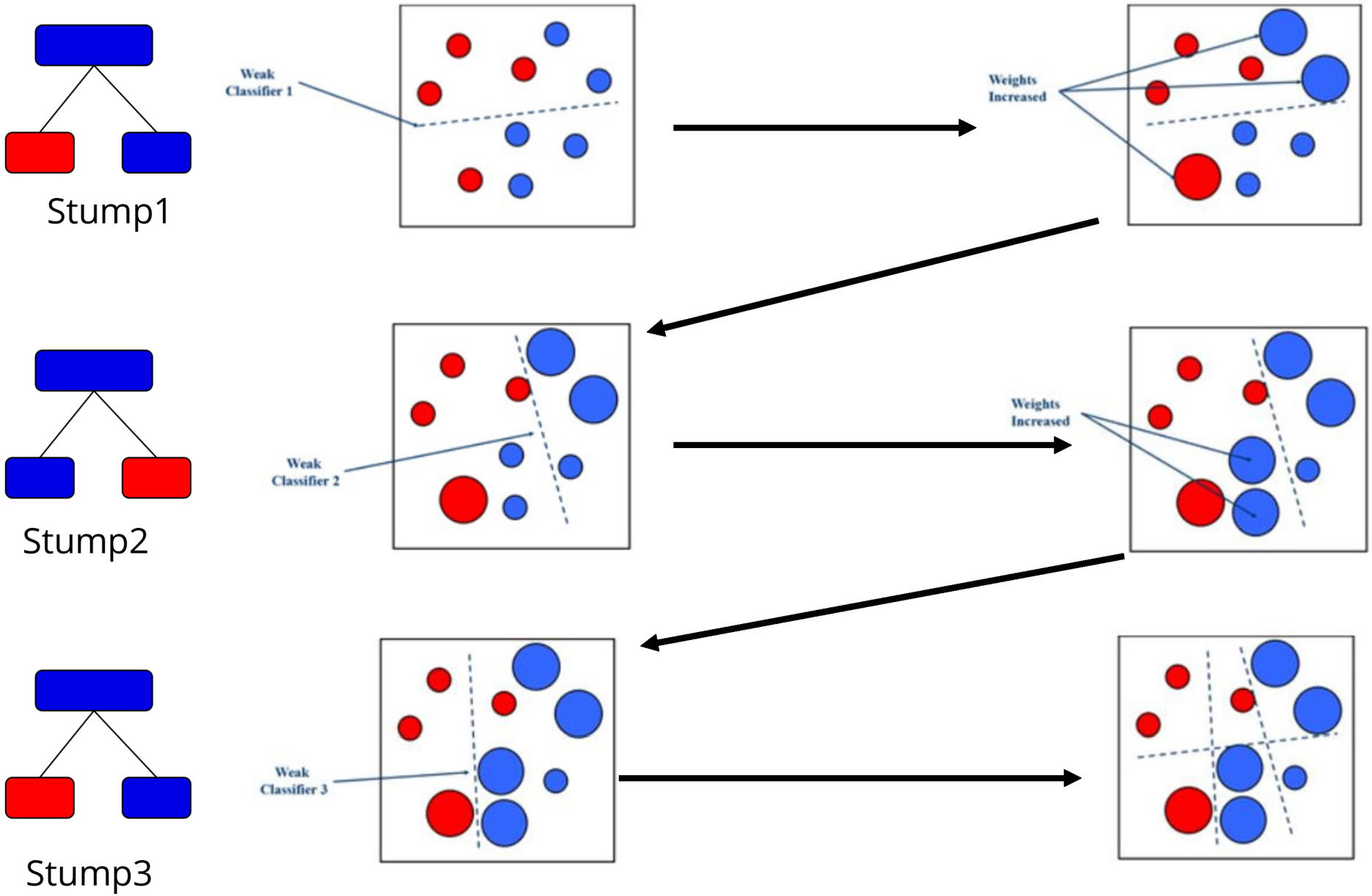


Weak Learners

Strong Learner

NB.Focus is on reducing bias

Boosting



Sample weights are updated after each iteration to increase the 'importance' of each misclassified record

Final Set of Learners

Boosting Ensemble

Feature	Adaptive Boosting	Gradient Tree Boosting	XGBoost	LightGBM	CatBoost
Learning Strategy	Weak learner ensemble (e.g., decision stumps)	Sequential decision trees	Regularized decision trees	Gradient-based one-sided sampling	Ordered boosting with categorical features
Objective Function	Any	Squared error, absolute error, log loss	Supports various objectives (customizable)	Supports various loss functions	Supports various loss functions (customizable)
Strength	Less prone to overfitting	Flexible with diverse loss functions	Highly efficient, customizable, regularized	Faster training, efficient with categorical features	Handles categorical features without one-hot encoding
Weakness	Sensitive to noisy data	Less efficient	More complex hyperparameter tuning	Less powerful than XGBoost	Requires more memory than LightGBM

Boosting & Bagging

Bagging	Boosting
Simplest way of combining predictions that belong to the same type.	A way of combining predictions that belong to the different types.
Aim to decrease variance, not bias.	Aim to decrease bias, not variance.
Each model receives equal weight.	Models are weighted according to their performance.
Each model is built independently.	New Model are influenced by performance of previously built models
if classifier is unstable (high variance) then apply bagging	if classifier is stable and simple(high bias) apply boosting
Tries to solve over-fitting problem.	Tries to reduce bias.