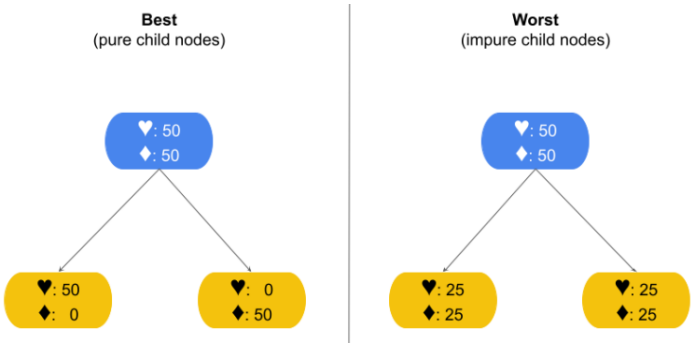
MODULE 4

Tree-based learning - Supervised machine learning that performs classification and regression tasks

Decision tree

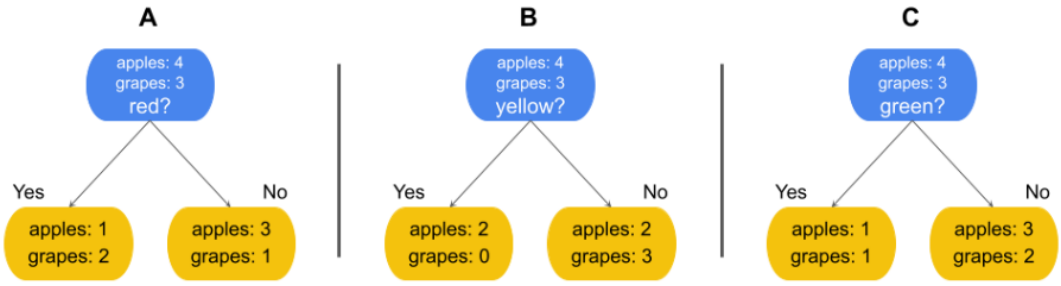
- Flowchart like structure that uses branching paths to predict the outcomes of event, probability of outcomes or to reach a decision.
- Can be used for classification problems and regression problems
- Types of nodes:
    - **Root node:** The first node of the tree where the first decision is made
    - **Decision node:** Nodes of the tree where decisions are made
    - **Leaf node:** Nodes where a final prediction is made
    - **Child node:** A node that results from a split
- **Impurity** - The degree of mixture with respect to class
    - Perfect split have no impurity (each child contains only a single class)
    - The split point that generate purest child nodes are selected to partition the data



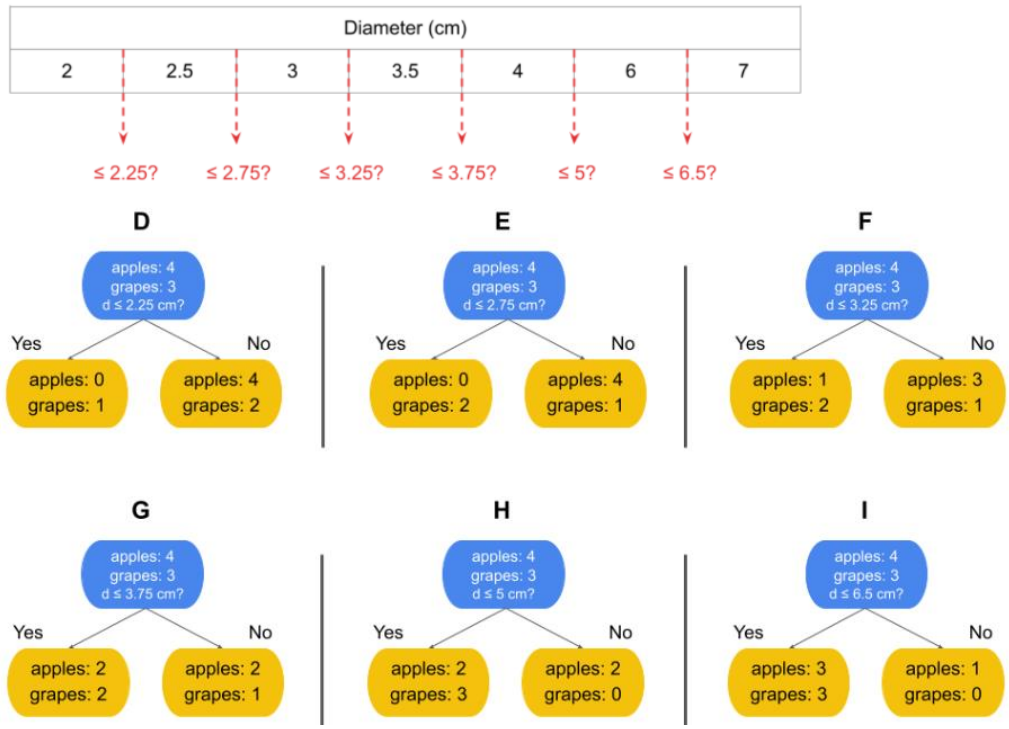
Example:

Color	Diameter (cm)	Fruit (target)
Yellow	3.5	Apple
Yellow	7	Apple
Red	2	Grape
Red	2.5	Grape
Green	4	Grape
Green	3	Apple
Red	6	Apple

First, the algorithm consider splitting based on categorical variable (color)



- If the predictor variable is continuous, the split can be made anywhere along the range of numbers that exist in the data.
- When dealing with large ranges of numbers, its common to consider split points based on percentiles of the distribution



Choosing splits

- Several metrics to use to determine the purity of a node to decide how to split:
  - **Gini impurity**
  - **Entropy**
  - **Information gain**
  - **Log loss**
- Gini impurity is the most straightforward metric and default for decision tree classifier in scikit-learn
- The best score are those that closest to 0.

$$Gini\ impurity\ =\ 1\ -\ \sum_{i=1}^{N} P(i)^2$$

$$Total\ Gini\ impurity\ =\ (\frac{n\ sample\ in\ left\ child}{total\ n\ samples\ in\ both\ child}) * Gi_{left}\ +\ (\frac{n\ samples\ in\ right\ child}{total\ n\ samples\ in\ both\ child}) Gi_{right}$$

Where i = class, $P(i)$ = probability of samples belonging to class i in a given mode

- The splitting will continue until all the leaf are pure or met a specified condition
- Advantages:
  - Requires few pre-processing steps
  - Can work easily with all types of variables (continuous, categorical, discrete)
  - Do not require normalization or scaling
  - Transparent decision

    - Not affected by extreme univariate values
- Disadvantages:
    - Can be computationally expensive relative to other algorithms
    - Small changes in data can result in significant changes in prediction

Hyperparameters - Parameters that can be set before the model is trained

Hyperparameter tuning – process of adjusting the parameters to find the best values that will result in the most optimal model

Max depth: max_depth
- defines how long a decision tree can get
- helps reduce overfitting and reduce co
- mputational complexity of training

Min samples split: min_samples_split
- minimum number of samples that a node must have for it to split into more nodes

Min samples leaf: min_samples_leaf
- defines the minimum number of samples for a leaf node

Grid search
- A technique that will train a model for every combination of preset ranges of hyperparameter values.
- The goal is to find the combination of values that results in a model that both fits the training data well.

Model validation
- Process of evaluating different models, selecting one, and then continuing to analyze the performance of the selected model to better understand its strengths and limitations.
- Split the dataset into three partition (train, validate & test)
    - Validation data is used instead of test data to compare different models
- Commonly used with very large dataset

Cross-validation (k-fold cross validation)
- Splits the training data into k number of folds, trains a model on k – 1 folds, and uses the fold that was held out to get a validation score. This process repeats k times
- Useful when working with smaller datasets (maximizes the data available)

Cross-validation (5-fold):

Score is not only the criteria to choose the best model, other factors should considered such as:
- How explainable is your model?
- How complex is it?
- How resilient is it against fluctuations in input values?
- How well does it perform on data not found in the training data?

- How much computational cost does it have to make predictions?
- Does it add much latency to any production system?

Ensemble learning (ensembling) – Builing multiple models and aggregating their predictions.

Base learner – each individual model that comprises an ensemble

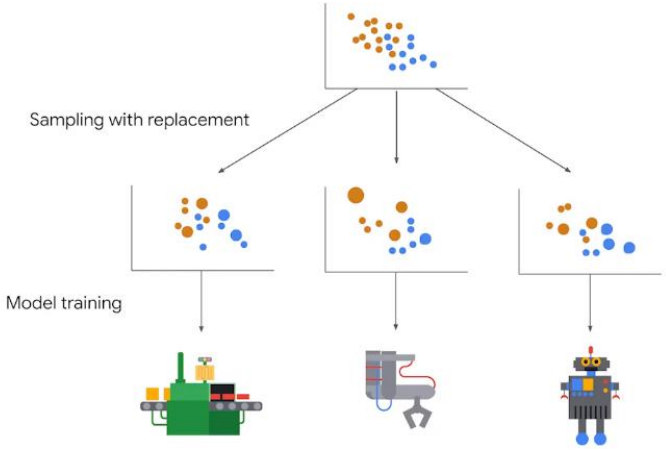
Weak learner – a model that performs slightly better than randomly guessing

Random forest – Ensemble of decision trees trained on bootstrapped data with randomly selected features

Bootstrapping – a resampling technique that helps in estimating the uncertainty of a statistical model

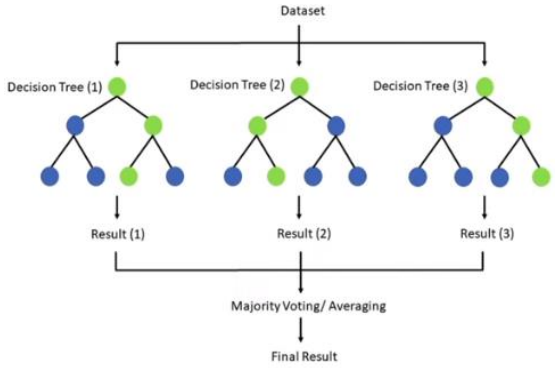
- Sampling the original dataset with replacement and generating multiple new datasets of the same size as the original
- Used to estimate the accuracy of a model, validate its performance and identify areas that need improvement

Bagging (Bootstrap aggregating) – Ensemble learning technique used to improve the stability and accuracy of machine learning models, particularly decision trees.



- Types of bagging:
    - **Homogeneous Bagging** – All base model use the same learning algorithms
    - **Heterogeneous Bagging** – Base models use different learning algorithms
- Reasons to use:
    - **Reduces variances** – Single model can result in high variance. Aggregating models predictions in ensemble helps to reduce it
    - **Fast** – training can happen in parallel across CPU cores and across different servers
    - **Good for big data** - Bagging doesn't require an entire training dataset to be stored in memory during training

Random Forest – An ensemble learning which combines the output of multiple decision trees with subset of features to reach a single result.



- Leverage randomness to reduce the likelihood base learner will make the same mistakes
- Bias and variance will reduced when the mistakes between learners are uncorrelated
- Hyperparameters:
    - **max_features** – specify the number of features that each tree randomly selects during training
    - **n_estimators** – specifies the number of trees the model will build in its ensemble

Boosting – Technique that builds an ensemble of weak learners sequentially, with each consecutive leaner trying to correct the errors of the one that preceded it

Adaptive boosting (AdaBoost)
A boosting methodology where each consecutive base learner assigns greater weight to the observations incorrectly predicted by the preceding leaner

Gradient boosting
A boosting methodology where each base learner in the sequence is built to predict the residual errors of the model that preceded it

```
learner1.fit(X, y)                  # Fit the data
ŷ1 = learner1.predict(X)            # Predict on X
error1 = y – ŷ1                     # Calculate the error → (actual – predicted)
learner2.fit(X, error1)             # Fit tree 2, but target = error from tree 1
ŷ2 = learner2.predict(X)            # Predict on X
error2 = ŷ2 – error1                # Calculate the new error
learner3.fit(X, error2)             # Fit tree 3, but target = error from tree 2
ŷ3 = learner3.predict(X)            # Predict on X
error3 = ŷ3 – error2                # Calculate the new error
```

Final prediction = learner1.predict(X) + learner2.predict(X) + learner3.predict(X)

- Ensemble models that use gradient boosting are called Gradient boosting machines (GBMs)
- **Advantages:**
    - High accuracy
    - Generally scalable
    - Works well with missing data
    - Don't require scaling
- **Disadvantages:**
    - Tuning many hyperparameters can be time consuming

- Difficult to interpret - do not have coefficients or directionality
- **Black-box model** – Any model whose predictions cannot be precisely explained
- Difficulty with **extrapolation** – Model's ability to predict new values that fall outside of the range of values in the training data
- Prone to overfitting if too many hyperparameters are tuned

XGBoost – Extreme gradient boosting, an optimized GBM package

Hyperparameters

- **learning_rate** (shrinkage)
  How much weight is given to each consecutive tree's prediction in the ensemble
  - Helps prevent over-correction and overfitting
  - Typical value are 0.01 – 0.03
- **min_child_weight**
  A tree will not split a node if it results in any child node with less weight than this value
  - The value in range of 0-1 interpreted in percentage and values above than 1 interpreted as no. of observations