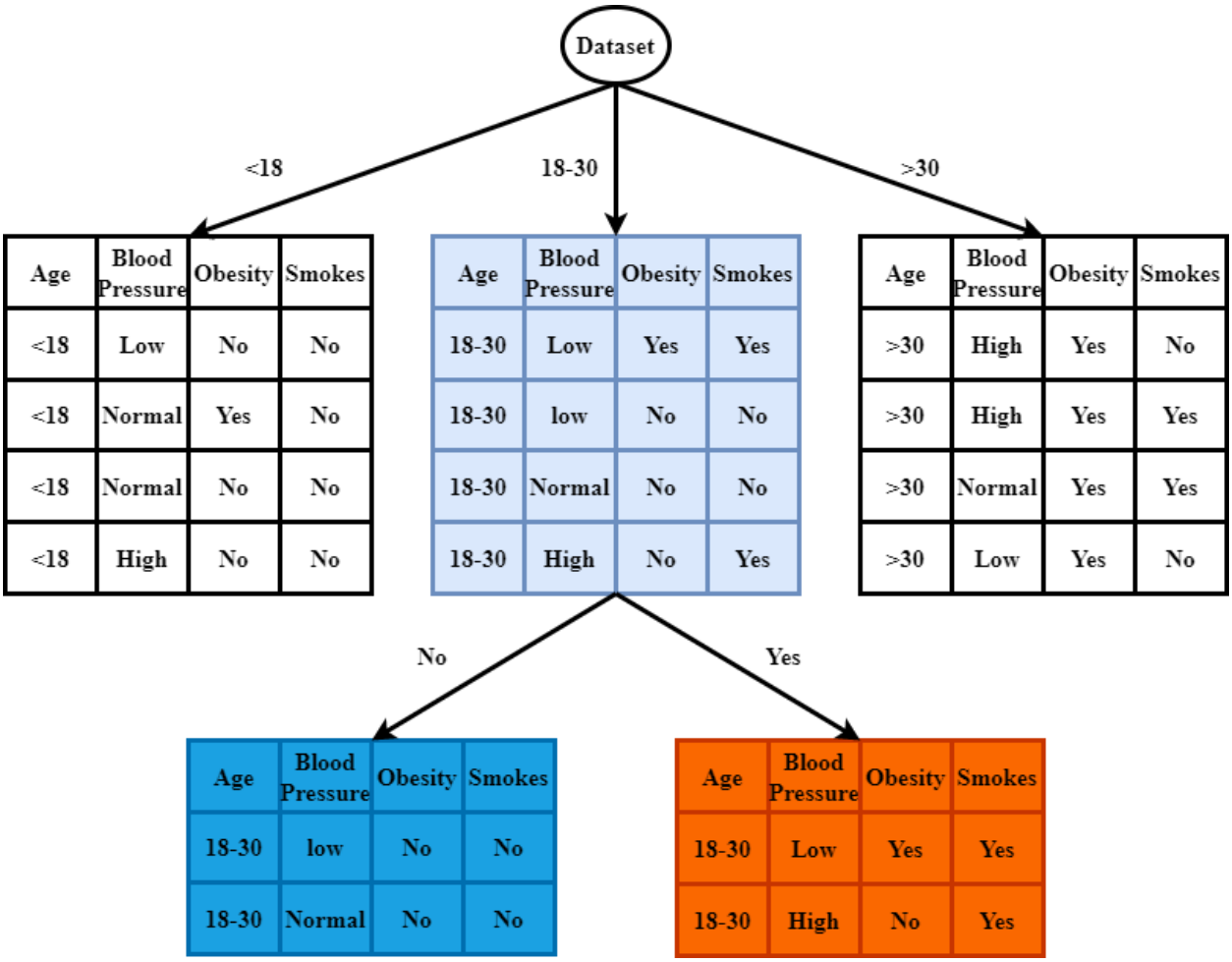
Big Data Classification

- **Overview**
  - Supervised learning methods to classify datasets.
  - Models to predict certain future behaviors, e.g., who is going to buy the latest phone?
  - Classification:
    - Predicts categorical class labels (discrete or nominal)
    - Classifies data (constructs a model) based on the training set and the values (class labels) in a classifying attribute and uses it in predicting class label for a new data.
  - Prediction:
    - Models continuous-valued functions, i.e., predicts unknown or missing values
  - Classification/Prediction methods:
    - Decision Tree based Methods
    - Rule-based Methods
    - Neural Networks
    - Naïve Bayes and Bayesian Belief Networks
    - Support Vector Machines
  - Typical Applications
    - User profile classification
    - Spam detection
    - Document categorization
    - Credit approval
    - Target marketing
    - Medical diagnosis

- **Decision Tree**
  - It is a classic data mining model.
  - Easy to implement and use.
  - An internal node is a test on an attribute.
  - At each node, one attribute is chosen to split training examples into distinct classes as much as possible.
    - Which attribute?
  - A branch represents an outcome of the test, e.g., Buy=”Yes”.
  - A leaf node represents a class label or class label distribution.
  - Classifying new data: A new data item is classified by following a matching path to a leaf node.
  - Example:

Age	Blood Pressure	Obesity	Smokes
>30	High	Yes	No
<18	Low	No	No
18-30	Low	Yes	Yes
<18	Normal	Yes	No
>30	High	Yes	Yes
18-30	Low	No	No
<18	Normal	No	No
>30	Normal	Yes	Yes
18-30	Normal	No	No
<18	High	No	No
18-30	High	No	Yes
>30	Low	Yes	No



Age	Blood Pressure	Obesity	Smokes
<18	Low	No	No
<18	Normal	Yes	No
<18	Normal	No	No
<18	High	No	No

Age	Blood Pressure	Obesity	Smokes
18-30	Low	Yes	Yes
18-30	low	No	No
18-30	Normal	No	No
18-30	High	No	Yes

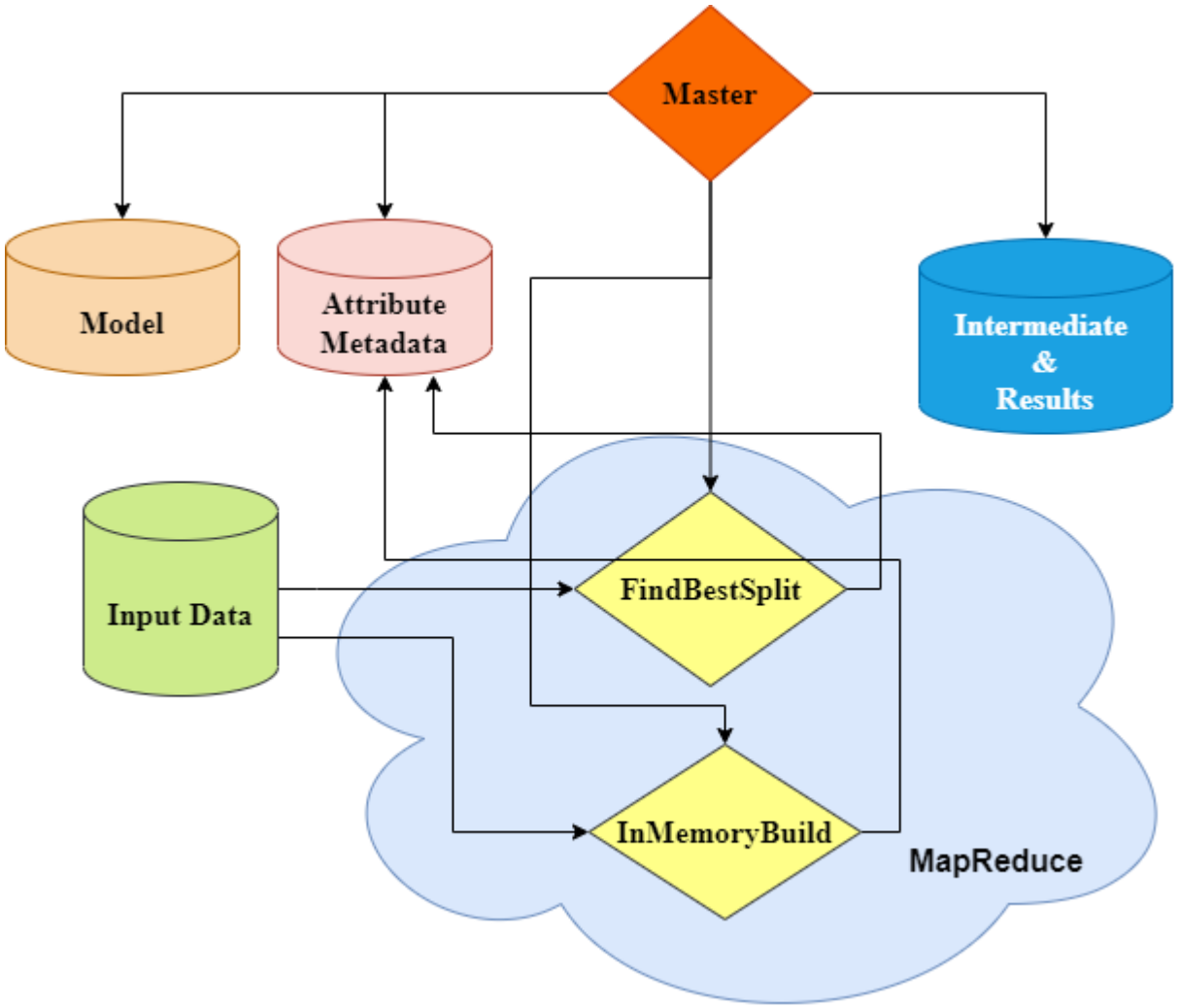
Age	Blood Pressure	Obesity	Smokes
>30	High	Yes	No
>30	High	Yes	Yes
>30	Normal	Yes	Yes
>30	Low	Yes	No

Age	Blood Pressure	Obesity	Smokes
18-30	low	No	No
18-30	Normal	No	No

Age	Blood Pressure	Obesity	Smokes
18-30	Low	Yes	Yes
18-30	High	No	Yes

- **PLANET Architecture**
  - **Overview**
    - Large dataset with hundreds of attributes
    - Dataset is too large to keep in memory and process on a single machine.
    - PLANET: **P**arallel **L**earner for **A**ssembling **N**umerous **E**nsemble **T**rees
      - First introduced in a Google research paper[Panda et al., VLDB ‘09].
    - PLANET is a learner for training decision trees that is built on MapReduce
    - Break up dataset across many processing units and then combine results.
    - It uses a sequence of MapReduce jobs to build a decision tree.
    - It builds the tree level by level.
    - General considerations:
      - Type of attributes: Hundreds of numerical (discrete and continuous) attributes
      - Class is numerical: Regression
      - Splits are binary
      - Decision tree is a binary tree and small enough for each mapper to keep in memory

- **PLANET Architecture Components**
  - **General Architecture:**



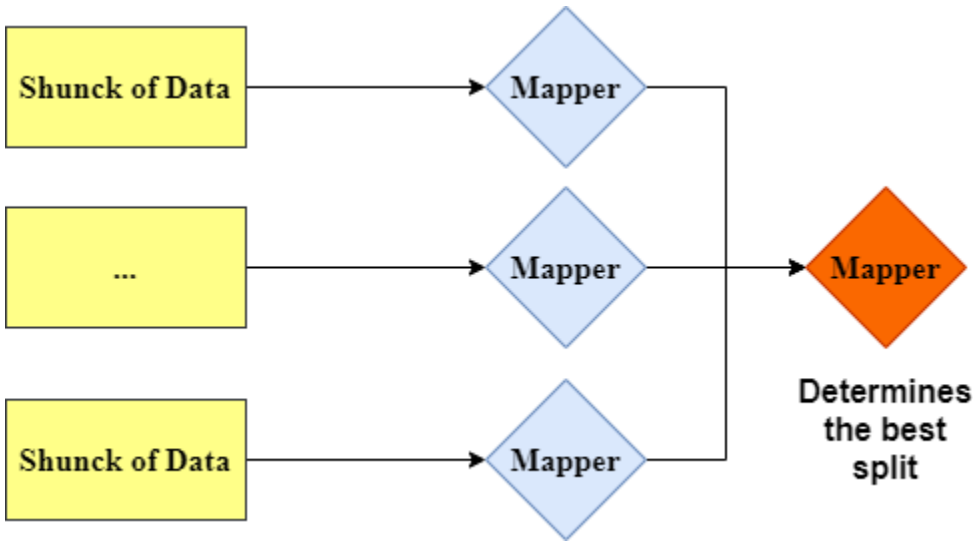
  - **Master:**
    - Monitors and controls everything, including running multiple MapReduce jobs.
    - It grows the tree for one level
    - Determines the state of the tree and grows it:
      - Decides if nodes should be split
      - If there is little data entering a node, runs an **InMemory-Build MapReduce** job to grow the entire subtree

        - For larger nodes, launches MapReduce **FindBestSplit** to find candidates for best split
        - Collects results from MapReduce jobs and chooses the best split for a node
        - Updates model
    - Master keeps two node queues:
        - MapReduceQueue (MRQ)
            - Nodes for which the dataset is too large to fit in memory
        - InMemoryQueue (InMemQ)
            - Nodes for which the dataset in the node fits in memory
- **Manage MapReduce processes**
- **Model File:**
    - A file describing the state of the model

- **MapReduce processes**
    - We build the tree level by level
    - One MapReduce step build one level of the tree: If we have 10 levels tree, we need 10 MapReduce operations.
    - Mapper:
        - Considers a number of possible splits (Xi,v) on its subset of data
        - For each split it stores partial statistics
        - Partial split-statistics is sent to Reducers
        - It loads the model and info about which attribute splits to consider
            - Each mapper sees a subset of the dataset
            - Mapper sends each data item to the appropriate leaf node L
            - For each leaf node L it keeps statistics about:

1. The data items reaching L
2. The data items in Left/Right subtree under split value

- Reducer:
  - Aggregates the statistics computed in the mapper step(The last two steps: (1) and (2).
  - It determines best split



- **Three types ofMapReduce jobs:**
  - **MapReduce Initialization (Run once)**
    - Identifies all the attribute values which need to be considered for splits.
    - Generates an “attribute metadata” to be loaded in memory by other tasks.
- **FindBestSplit MapReduce: Run multiple times**
  - MapReduce job to find best split when there is too much data to fit in memory.
- **InMemoryBuild MapReduce: Run once last**

- Task to grow an entire subtree once the data for it fits in memory
- Grows an entire sub-tree once the data fits in memory