Professur Technische Informatik
Prof. Dr. Wolfram Hardt

Python: decision tree

Exercise 1.10

Shadi Saleh

Agenda

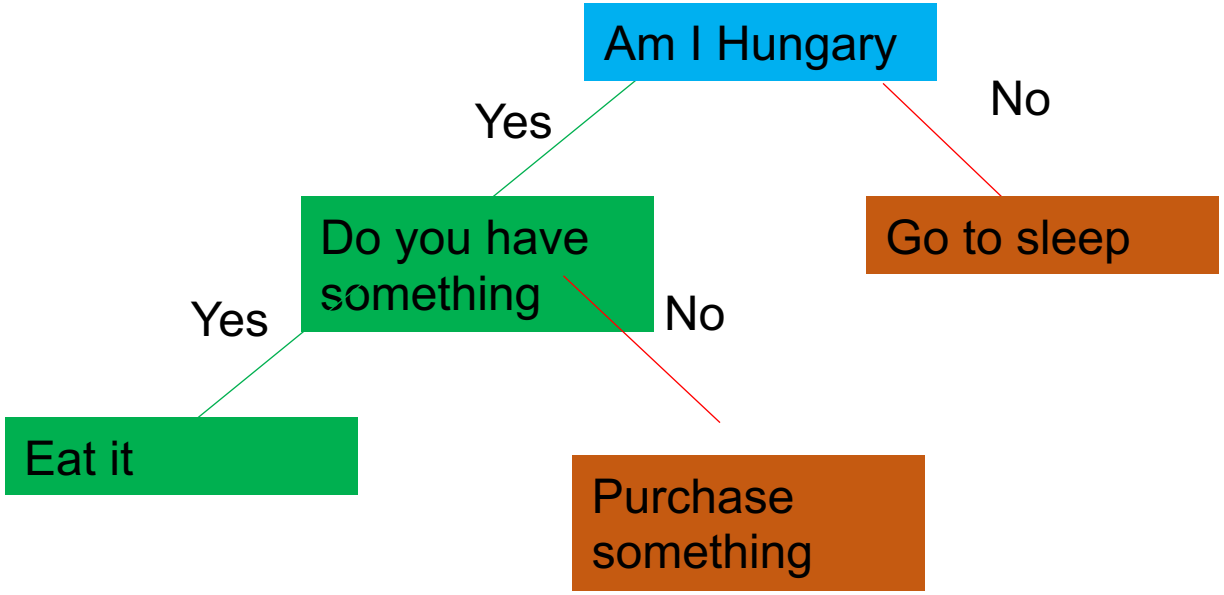
- My Experience
- AI is The future
- Conventional (Passive) Deep Learning
- Active Deep Learning
- Our last AI activities

Decision tree

- Decision Tree algorithm belongs to the family of supervised learning algorithms.
- Unlike other supervised learning algorithms, the decision tree algorithm can be used for solving **regression and classification problems** too.
- The goal of using a Decision Tree is to create a training model that can use to predict the class or value of the target variable by **learning simple decision rules** inferred from prior data(training data).

Decision tree

- Graphical representation of all the possible solutions to a decision
- Decisions are made on some conditions
- Decisions made can be easily explained

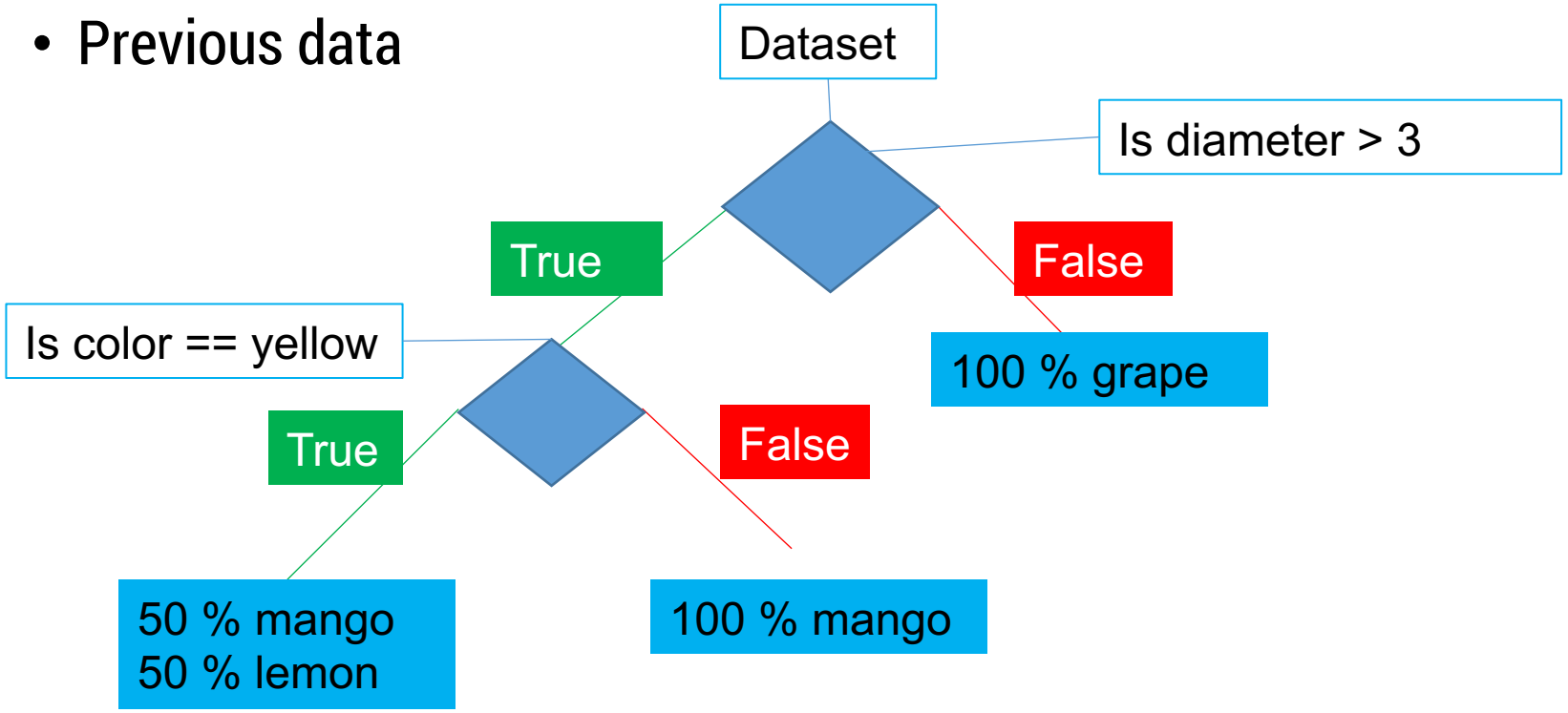


Dataset

Color	Diameter	Label
Green	3	Mango
Yellow	3	Mango
Red	1	Grape
Red	1	Grape
Yellow	3	Lemon

Example

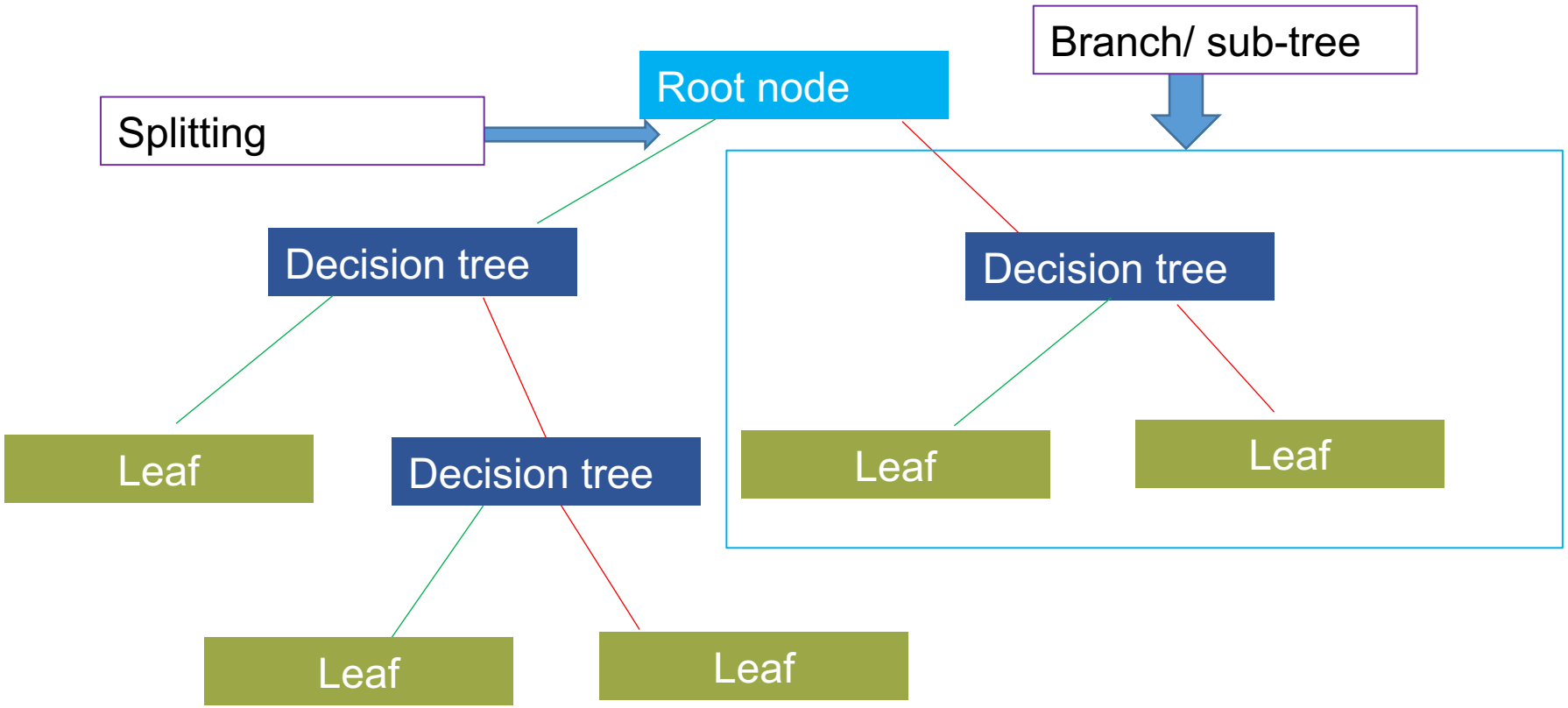
- Previous data



Terminologies: Decision Trees

1. **Root Node:** It represents the entire population or sample and this further gets divided into two or more homogeneous sets.
2. **Splitting:** A process of dividing a node into two or more sub-nodes.
3. **Decision Node:** When a sub-node splits into further sub-nodes
4. **Leaf / Terminal Node:** Nodes do not split is called Leaf
5. **Pruning:** When we remove sub-nodes of a decision node, this process is called pruning. You can say the opposite process of splitting.
6. **Branch / Sub-Tree:** A subsection of the entire tree is called branch
7. **Parent and Child Node:** A node, which is divided into sub-nodes is called are called child of that parent

Terminonologies



Decision tree: phases

- Two phases in decision tree
  - Building phase
    - Preprocess the dataset
    - Split the dataset from train and test using python
    - Train the classifier
  - Operational phase
    - Make prediction
    - Calculate the accuracy

Data import and Data slicing

- **Data Import :**
  - To import and manipulate the data, *pandas* package is available
  - As the dataset is separated by "," so we have to pass the sep parameter's value as ",".
- **Data Slicing :**
  - Before training split the dataset into the training and testing dataset.
  - To split the dataset for training and testing we are using the sklearn module *train_test_split*
  - Firstly separate the target variable from the attributes in the dataset

```
X = balance_data.values[:, 1:5]
Y = balance_data.values[:,0]
```

Splitting data

- To understand model performance, dividing the dataset into a training set and a test set is a good strategy.
- Let's split the dataset by using function train_test_split(). You need to pass 3 parameters features, target, and test_set size.
- Split the dataset for training and testing purpose.

  – X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.3, random_state = 100)

Building decision tree model

- create a Decision Tree Model using Scikit-learn.
- # Create Decision Tree classifer object
- clf = DecisionTreeClassifier()
- # Train Decision Tree Classifer
- clf = clf.fit(X_train,y_train)
- #Predict the response for test dataset
- y_pred = clf.predict(X_test)

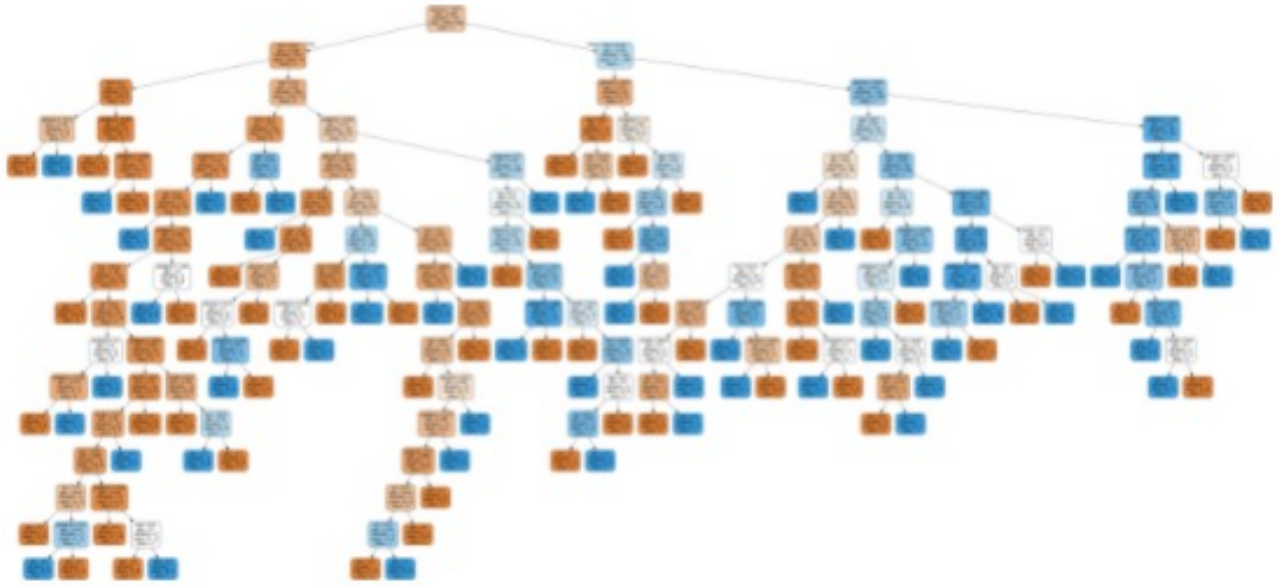
Evaluating model

- Let's estimate, how accurately the classifier or model can predict the type of cultivars.
- Accuracy can be computed by comparing actual test set values and predicted values.
- # Model Accuracy, how often is the classifier correct?
  print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

- Output
  - Accuracy: 0.6753246753246753

Visualizing Decision Trees

- You can use Scikit-learn's *export_graphviz* function for display the tree within a Jupyter notebook. For plotting tree, you also need to install graphviz and pydotplus.



Optimizing Decision Tree Performance

- **Criterion:**
  - **optional (default="gini") or Choose attribute selection measure**: This parameter allows us to use the different-different attribute selection measure. Supported criteria are "gini" for the Gini index and "entropy" for the information gain.
- **Splitter :**
  - **string, optional (default="best") or Split Strategy**: This parameter allows us to choose the split strategy. Supported strategies are "best" to choose the best split and "random" to choose the best random split.
- **max_depth :**
  - **int or None, optional (default=None) or Maximum Depth of a Tree**: The maximum depth of the tree. If None, then nodes are expanded until all the leaves contain less than min_samples_split samples. The higher value of maximum depth causes overfitting, and a lower value causes underfitting (Source).

Pros

- Decision trees are easy to interpret and visualize.
- It can easily capture Non-linear patterns.
- It requires fewer data preprocessing from the user, for example, there is no need to normalize columns.
- It can be used for feature engineering such as predicting missing values, suitable for variable selection.
- The decision tree has no assumptions about distribution because of the non-parametric nature of the algorithm. (Source)

Cons

- Sensitive to noisy data. It can overfit noisy data.
- The small variation(or variance) in data can result in the different decision tree. This can be reduced by bagging and boosting algorithms.
- Decision trees are biased with imbalance dataset, so it is recommended that balance out the dataset before creating the decision tree.

Exercise

- Perform the similar task as discussed in the lecture on different dataset
  - Load data
  - Feature selection
  - Splitting data
  - Building decision tree
  - Evaluating model

Thank you for your attention

Shadi Saleh

Professur Technische Informatik
Fakultät für Informatik

Technische Universität Chemnitz
Straße der Nationen 62
09111 Chemnitz
Germany

shadi.saleh@informatik.tu-chemnitz.de