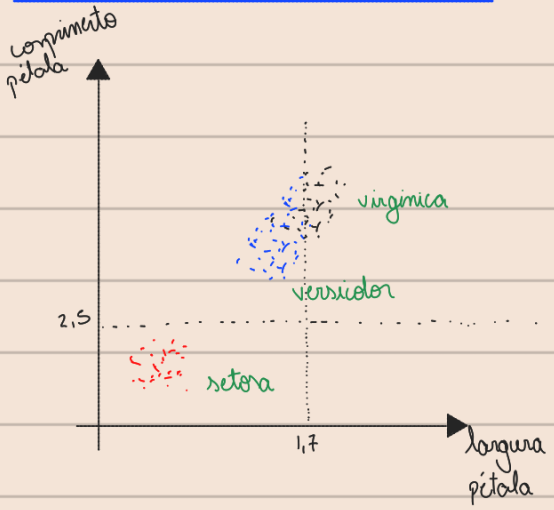
Decision Tree

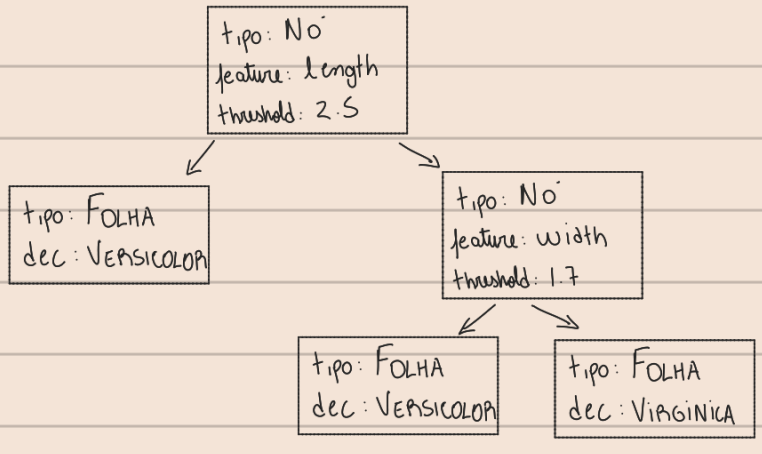



```
def classifica (lenght, width):
    if lenght < 2,5:
        return "setosa"
    else
        if width < 1,7:
            return "versicolor"
        else
            return "virginica"
```

Algoritmo

```
Decision Tree (dataset):    (TREINO)
    se nao da_para_dividir (dataset):
        nó.tipo = "FOLHA"
        nó.decisão = constroi_decisão (dataset)
        return nó
    feature, threshold = acha_separação (dataset)
    dataset_esq, dataset_dir = separa (dataset, feature, threshold)
    nó.esq = Decision Tree (dataset_esq)
    nó.dir = Decision Tree (dataset_dir)
    nó.tipo = "NÓ"
    nó.feature = feature
    nó.threshold = threshold
    nó.ramo_esq = nó_esq
    nó.ramo_dir = nó_esq
    return nó
```

Teste de mesa




```
Decision Tree.predict (raiz, amostra):
    se raiz.tipo == "FOLHA"
        return raiz.decisão
    se amostra[raiz.feature] < raiz.threshold:
```

return DecisionTree.predict(raiz.nó_esq, amostra)
return DecisionTree.predict(raiz.nó_dir, amostra)

Vantagens
* não precisa usar StandardScaler antes
* intrinsicamente multiclasse
  - retorno em prob
  - ordem dos cortes traz importância das features pela altura

Desvantagens
* custo de treinamento
* não é invariante a rotação