Árvores de Decisão do Zero: ID3, C4.5 e CART (Titanic) · Documento Único

Autora: Laura Araújo

Data: 24/09/2025

Observações do enunciado

1. Discussões, código e resultados **neste único PDF**.

2. O código está em pacote instalável `dtree_lab/`.

3. Execução com *runner* (opcional) e scripts descritos neste documento.

Seção 1 · Preparação dos dados (Titanic)

Atributos: `Pclass`, `Sex`, `Age`, `SibSp`, `Parch`, `Fare`, `Embarked`.

Tratamento: imputação (mediana/moda), *split* 80/20 estratificado, discretização (`Age`, `Fare`) por quantis para ID3.

Código · utilitários (imputação, split, discretização)

```
from __future__ import annotations
from typing import Tuple
import numpy as np
import pandas as pd

def train_test_split_stratified(y: np.ndarray, test_size: float = 0.2, seed: int = 42) -> Tup
    rng = np.random.default_rng(seed)
    idx = np.arange(len(y))
    test_idx = []
    for c in np.unique(y):
        class_idx = idx[y == c]
        rng.shuffle(class_idx)
        n_test = max(1, int(round(test_size * len(class_idx))))
        test_idx.extend(class_idx[:n_test])
    test_idx = np.array(sorted(test_idx))
    train_idx = np.array([i for i in idx if i not in set(test_idx)])
    return train_idx, test_idx


def discretize_equal_frequency(series: pd.Series, bins: int = 4, labels: bool = True) -> pd.S
    """Discretiza por quantis (~mesmo número de amostras por faixa)."""
    q = np.linspace(0, 1, bins + 1)
    edges = np.unique(series.quantile(q).values)
    edges[0] = -np.inf
    edges[-1] = np.inf
    cats = pd.cut(series, bins=edges, include_lowest=True)
    return cats.astype(str) if labels else cats


def discretize_equal_width(series: pd.Series, bins: int = 4, labels: bool = True) -> pd.Serie
    """Discretiza por largura fixa (intervalos iguais)."""
    cats = pd.cut(series, bins=bins, include_lowest=True)
    return cats.astype(str) if labels else cats


def impute_simple(df: pd.DataFrame) -> pd.DataFrame:
    """Imputa NaNs: numéricos -> mediana; categóricos/objeto -> moda."""
    out = df.copy()
    for col in out.columns:
        if out[col].dtype == object:
            if out[col].isna().any():
                out[col] = out[col].fillna(out[col].mode().iloc[0])
        else:
            if out[col].isna().any():
                out[col] = out[col].fillna(out[col].median())
    return out
```

Métricas auxiliares

```python
from __future__ import annotations
import numpy as np

def accuracy(y_true, y_pred) -> float:
    y_true = np.asarray(y_true)
    y_pred = np.asarray(y_pred)
    return float((y_true == y_pred).mean())


def confusion_matrix(y_true, y_pred):
    y_true = np.asarray(y_true)
    y_pred = np.asarray(y_pred)
    labels = sorted(list(set(y_true) | set(y_pred)))
    L = len(labels)
    lab2i = {lab: i for i, lab in enumerate(labels)}
    m = np.zeros((L, L), dtype=int)
    for t, p in zip(y_true, y_pred):
        m[lab2i[t], lab2i[p]] += 1
    return labels, m
```

Seção 2 · Implementações do zero

2.1 ID3 (ganho de informação; atributos categóricos)

```python
from __future__ import annotations
from dataclasses import dataclass
from typing import Any, Dict, List, Optional
import numpy as np
import pandas as pd
from collections import Counter


def _entropy(y: np.ndarray) -> float:
    vals, cnt = np.unique(y, return_counts=True)
    if cnt.sum() == 0:
        return 0.0
    p = cnt / cnt.sum()
    return float(-(p * np.log2(p)).sum())


def _info_gain_categorical(x: np.ndarray, y: np.ndarray) -> float:
    ent_total = _entropy(y)
    vals, cnt = np.unique(x, return_counts=True)
    ent_cond = 0.0
    for v, c in zip(vals, cnt):
        ent_cond += (c / len(x)) * _entropy(y[x == v])
    return ent_total - ent_cond

@dataclass
class ID3Node:
    atributo: Optional[str] = None
    filhos: Optional[Dict[Any, 'ID3Node']] = None
    rotulo: Optional[Any] = None
    is_leaf: bool = False
    classe_majoritaria: Optional[Any] = None

    def __str__(self, nivel: int = 0) -> str:
        ident = " " * nivel
        if self.is_leaf:
            return f"""{ident}Folha: {self.rotulo}
"""
        s = f"""{ident}[{self.atributo}]
"""
        for v, fchild in (self.filhos or {}).items():
            s += f"""{ident} -> {v}:
{fchild.__str__(nivel+2)}"""
        return s

class ID3:
    def __init__(self, max_depth: Optional[int] = None, min_samples_split: int = 2):
        self.max_depth = max_depth
        self.min_samples_split = min_samples_split
        self.tree_: Optional[ID3Node] = None
        self.features_: List[str] = []

    def _best_attribute(self, X: pd.DataFrame, y: np.ndarray, attrs: List[str]) -> Optional[s
        gains = []
        for col in attrs:
            gains.append((_info_gain_categorical(X[col].values, y), col))
        gains.sort(key=lambda t: (-t[0], X[t[1]].nunique(), t[1]))
        return gains[0][1] if gains else None
```

2.2 C4.5 (razão de ganho; contínuos por limiar; categórico multi-ramo)

```
from __future__ import annotations
from dataclasses import dataclass
from typing import Any, Dict, List, Optional, Tuple
from collections import Counter
import numpy as np
import pandas as pd


def _entropy_list(y_list: List[Any]) -> float:
    cnt = Counter(y_list); n = len(y_list)
    return -sum((c/n)*np.log2(c/n) for c in cnt.values()) if n else 0.0


def _gain_ratio_numeric(col: List[float], y: List[Any], thr: float) -> Tuple[float, float]:
    left_y = [y[i] for i in range(len(y)) if col[i] < thr]
    right_y = [y[i] for i in range(len(y)) if col[i] >= thr]
    if not left_y or not right_y:
        return 0.0, 0.0
    ent_total = _entropy_list(y)
    pL = len(left_y)/len(y); pR = 1.0 - pL
    ent_div = pL * _entropy_list(left_y) + pR * _entropy_list(right_y)
    info_gain = ent_total - ent_div
    split_info = -sum(p * np.log2(p) for p in [pL, pR] if p > 0)
    return info_gain, (info_gain/split_info if split_info != 0 else 0.0)


def _best_split_numeric(col: List[float], y: List[Any]) -> Tuple[Optional[float], float]:
    uniq = sorted(set(col))
    if len(uniq) < 2:
        return None, -1.0
    cands = [(uniq[i] + uniq[i+1]) / 2 for i in range(len(uniq) - 1)]
    best_thr, best_gr = None, -1.0
    for t in cands:
        _, gr = _gain_ratio_numeric(col, y, t)
        if gr > best_gr:
            best_gr, best_thr = gr, t
    return best_thr, best_gr

@dataclass
class C45Node:
    atributo: Optional[str] = None
    limiar: Optional[float] = None
    filhos: Optional[Dict[Any, 'C45Node']] = None
    rotulo: Optional[Any] = None
    is_leaf: bool = False

    def __str__(self, nivel: int = 0) -> str:
        ident = " " * nivel
        if self.is_leaf:
            return f"""{ident}Folha: {self.rotulo}
"""
        if self.limiar is not None:
            s = f"""{ident}[{self.atributo} < {self.limiar:.6g}]
"""
            s += f"""{ident} -> < :
{self.filhos['<'].__str__(nivel+2)}"""
            s += f"""{ident} -> >=:
```

2.3 CART (índice de Gini; divisões binárias; categórico por subconjunto)

```
from __future__ import annotations
from dataclasses import dataclass
from typing import Any, Dict, List, Optional, Set, Tuple
from collections import Counter
import numpy as np
import pandas as pd


def _gini(y) -> float:
    y = np.asarray(y); n = len(y)
    if n == 0: return 0.0
    cnt = Counter(y)
    return 1.0 - sum((c/n)**2 for c in cnt.values())

@dataclass
class CARTNode:
    atributo: Optional[str] = None
    limiar: Optional[float] = None
    cats_esq: Optional[Set[Any]] = None
    is_categorical: bool = False
    filhos: Optional[Dict[str, 'CARTNode']] = None
    rotulo: Optional[Any] = None
    is_leaf: bool = False

    def __str__(self, nivel: int = 0) -> str:
        ident = " " * nivel
        if self.is_leaf:
            return f"""{ident}Folha: {self.rotulo}
"""
        if self.is_categorical:
            s = f"""{ident}[{self.atributo} · {sorted(list(self.cats_esq or []))}]
"""
            s += f"""{ident} -> · :
{self.filhos['in'].__str__(nivel+2)}"""
            s += f"""{ident} -> · :
{self.filhos['out'].__str__(nivel+2)}"""
            return s
        s = f"""{ident}[{self.atributo} < {self.limiar:.6g}]
"""
        s += f"""{ident} -> < :
{self.filhos['<'].__str__(nivel+2)}"""
        s += f"""{ident} -> >=:
{self.filhos['>='].__str__(nivel+2)}"""
        return s

class CART:
    def __init__(self, max_depth: Optional[int] = None, min_samples_split: int = 2, min_sampl
        self.max_depth = max_depth
        self.min_samples_split = min_samples_split
        self.min_samples_leaf = min_samples_leaf
        self.tree_: Optional[CARTNode] = None
        self.features_: List[str] = []
        self.cat_features_: Set[str] = set()

    def _best_split_numeric(self, x: np.ndarray, y: np.ndarray) -> Tuple[Optional[float], flo
        uniq = np.unique(x)
        if uniq.size < 2: return None, np.inf
```

Seção 3 · Resultados com o `train.csv` enviado

Configuração: *split* estratificado 80/20 (`seed=42`), `max_depth=6`.

ID3

Acurácia (treino): 0.8808 \nAcurácia (teste): 0.8034

Matriz de confusão (teste):

labels: 0, 1

||0|1|

|-|-|-|

|0|99|11|

|1|24|44|

Árvore (ID3)

```
[Sex]
 -> female:
  [Pclass]
   -> 1:
    [Fare]
     -> (14.454, 30.5]:
      [Embarked]
       -> C:
        [Age]
         -> (35.0, inf]:
          [Parch]
           -> 0:
            Folha: 0
       -> S:
        Folha: 1
     -> (30.5, inf]:
      Folha: 1
   -> 2:
    [Age]
     -> (-inf, 22.0]:
      Folha: 1
     -> (22.0, 28.0]:
      [Parch]
       -> 0:
        [Embarked]
         -> C:
          Folha: 1
         -> S:
          [Fare]
           -> (14.454, 30.5]:
            Folha: 1
           -> (30.5, inf]:
            Folha: 1
           -> (7.896, 14.454]:
            Folha: 1
       -> 1:
        [SibSp]
         -> 0:
          Folha: 1
         -> 1:
          [Embarked]
           -> S:
            Folha: 0
         -> 2:
          Folha: 1
       -> 2:
        Folha: 1
       -> 3:
        Folha: 1
     -> (28.0, 35.0]:
      Folha: 1
     -> (35.0, inf]:
      [Parch]
       -> 0:
        [Fare]
         -> (14.454, 30.5]:
          [Embarked]
           -> S:
```

Regras (ID3)

```
Sex == female AND Pclass == 1 AND Fare == (14.454, 30.5] AND Embarked == C AND Age == (35.0,
Sex == female AND Pclass == 1 AND Fare == (14.454, 30.5] AND Embarked == S => predict 1
Sex == female AND Pclass == 1 AND Fare == (30.5, inf] => predict 1
Sex == female AND Pclass == 2 AND Age == (-inf, 22.0] => predict 1
Sex == female AND Pclass == 2 AND Age == (22.0, 28.0] AND Parch == 0 AND Embarked == C => pre
Sex == female AND Pclass == 2 AND Age == (22.0, 28.0] AND Parch == 0 AND Embarked == S AND Fa
Sex == female AND Pclass == 2 AND Age == (22.0, 28.0] AND Parch == 0 AND Embarked == S AND Fa
Sex == female AND Pclass == 2 AND Age == (22.0, 28.0] AND Parch == 0 AND Embarked == S AND Fa
Sex == female AND Pclass == 2 AND Age == (22.0, 28.0] AND Parch == 1 AND SibSp == 0 => predic
Sex == female AND Pclass == 2 AND Age == (22.0, 28.0] AND Parch == 1 AND SibSp == 1 AND Embar
Sex == female AND Pclass == 2 AND Age == (22.0, 28.0] AND Parch == 1 AND SibSp == 2 => predic
Sex == female AND Pclass == 2 AND Age == (22.0, 28.0] AND Parch == 2 => predict 1
Sex == female AND Pclass == 2 AND Age == (22.0, 28.0] AND Parch == 3 => predict 1
Sex == female AND Pclass == 2 AND Age == (28.0, 35.0] => predict 1
Sex == female AND Pclass == 2 AND Age == (35.0, inf] AND Parch == 0 AND Fare == (14.454, 30.5
Sex == female AND Pclass == 2 AND Age == (35.0, inf] AND Parch == 0 AND Fare == (7.896, 14.45
Sex == female AND Pclass == 2 AND Age == (35.0, inf] AND Parch == 1 => predict 1
Sex == female AND Pclass == 3 AND SibSp == 0 AND Age == (-inf, 22.0] AND Parch == 0 AND Fare
Sex == female AND Pclass == 3 AND SibSp == 0 AND Age == (-inf, 22.0] AND Parch == 0 AND Fare
Sex == female AND Pclass == 3 AND SibSp == 0 AND Age == (-inf, 22.0] AND Parch == 0 AND Fare
Sex == female AND Pclass == 3 AND SibSp == 0 AND Age == (-inf, 22.0] AND Parch == 1 => predic
Sex == female AND Pclass == 3 AND SibSp == 0 AND Age == (-inf, 22.0] AND Parch == 2 AND Embar
Sex == female AND Pclass == 3 AND SibSp == 0 AND Age == (-inf, 22.0] AND Parch == 2 AND Embar
Sex == female AND Pclass == 3 AND SibSp == 0 AND Age == (22.0, 28.0] AND Embarked == C AND Pa
Sex == female AND Pclass == 3 AND SibSp == 0 AND Age == (22.0, 28.0] AND Embarked == C AND Pa
Sex == female AND Pclass == 3 AND SibSp == 0 AND Age == (22.0, 28.0] AND Embarked == C AND Pa
Sex == female AND Pclass == 3 AND SibSp == 0 AND Age == (22.0, 28.0] AND Embarked == Q AND Fa
Sex == female AND Pclass == 3 AND SibSp == 0 AND Age == (22.0, 28.0] AND Embarked == Q AND Fa
Sex == female AND Pclass == 3 AND SibSp == 0 AND Age == (22.0, 28.0] AND Embarked == S AND Pa
Sex == female AND Pclass == 3 AND SibSp == 0 AND Age == (22.0, 28.0] AND Embarked == S AND Pa
Sex == female AND Pclass == 3 AND SibSp == 0 AND Age == (22.0, 28.0] AND Embarked == S AND Pa
Sex == female AND Pclass == 3 AND SibSp == 0 AND Age == (28.0, 35.0] AND Fare == (-inf, 7.896
Sex == female AND Pclass == 3 AND SibSp == 0 AND Age == (28.0, 35.0] AND Fare == (14.454, 30.
Sex == female AND Pclass == 3 AND SibSp == 0 AND Age == (28.0, 35.0] AND Fare == (7.896, 14.4
Sex == female AND Pclass == 3 AND SibSp == 0 AND Age == (35.0, inf] AND Embarked == C => pred
Sex == female AND Pclass == 3 AND SibSp == 0 AND Age == (35.0, inf] AND Embarked == Q => pred
Sex == female AND Pclass == 3 AND SibSp == 0 AND Age == (35.0, inf] AND Embarked == S AND Far
Sex == female AND Pclass == 3 AND SibSp == 0 AND Age == (35.0, inf] AND Embarked == S AND Far
Sex == female AND Pclass == 3 AND SibSp == 1 AND Parch == 0 AND Embarked == C AND Age == (-in
Sex == female AND Pclass == 3 AND SibSp == 1 AND Parch == 0 AND Embarked == C AND Age == (22.
Sex == female AND Pclass == 3 AND SibSp == 1 AND Parch == 0 AND Embarked == Q => predict 1
Sex == female AND Pclass == 3 AND SibSp == 1 AND Parch == 0 AND Embarked == S AND Fare == (-i
Sex == female AND Pclass == 3 AND SibSp == 1 AND Parch == 0 AND Embarked == S AND Fare == (14
Sex == female AND Pclass == 3 AND SibSp == 1 AND Parch == 0 AND Embarked == S AND Fare == (7.
Sex == female AND Pclass == 3 AND SibSp == 1 AND Parch == 1 AND Embarked == C => predict 0
Sex == female AND Pclass == 3 AND SibSp == 1 AND Parch == 1 AND Embarked == Q => predict 0
Sex == female AND Pclass == 3 AND SibSp == 1 AND Parch == 1 AND Embarked == S AND Age == (-in
Sex == female AND Pclass == 3 AND SibSp == 1 AND Parch == 1 AND Embarked == S AND Age == (22.
Sex == female AND Pclass == 3 AND SibSp == 1 AND Parch == 1 AND Embarked == S AND Age == (28.
Sex == female AND Pclass == 3 AND SibSp == 1 AND Parch == 2 => predict 0
Sex == female AND Pclass == 3 AND SibSp == 1 AND Parch == 3 => predict 0
Sex == female AND Pclass == 3 AND SibSp == 1 AND Parch == 4 => predict 0
Sex == female AND Pclass == 3 AND SibSp == 1 AND Parch == 5 AND Age == (35.0, inf] AND Embark
Sex == female AND Pclass == 3 AND SibSp == 1 AND Parch == 6 => predict 0
Sex == female AND Pclass == 3 AND SibSp == 2 AND Embarked == C => predict 1
Sex == female AND Pclass == 3 AND SibSp == 2 AND Embarked == S => predict 0
Sex == female AND Pclass == 3 AND SibSp == 3 => predict 0
Sex == female AND Pclass == 3 AND SibSp == 4 AND Fare == (30.5, inf] AND Age == (-inf, 22.0]
```

C45

Acurácia (treino): 0.8219 \nAcurácia (teste): 0.8202

Matriz de confusão (teste):

labels: 0, 1

||0|1|

|-|-|-|

|0|103|7|

|1|25|43|

Árvore (C45)

```
[Sex]
 -> female:
  [SibSp < 6]
   -> < :
    [Pclass < 2.5]
     -> < :
      [Fare < 28.8562]
       -> < :
        [Age < 53.5]
         -> < :
          [Parch < 1.5]
           -> < :
            Folha: 1
           -> >=:
            Folha: 1
         -> >=:
          [Embarked]
           -> S:
            Folha: 1
       -> >=:
        Folha: 1
     -> >=:
      [Fare < 32.8813]
       -> < :
        [Age < 1.5]
         -> < :
          Folha: 1
         -> >=:
          [Embarked]
           -> C:
            Folha: 1
           -> Q:
            Folha: 1
           -> S:
            Folha: 0
       -> >=:
        Folha: 0
   -> >=:
    Folha: 0
 -> male:
  [Age < 1.5]
   -> < :
    Folha: 1
   -> >=:
    [Fare < 387.665]
     -> < :
      [SibSp < 4.5]
       -> < :
        [Parch < 2.5]
         -> < :
          [Pclass < 1.5]
           -> < :
            Folha: 0
           -> >=:
            Folha: 0
         -> >=:
          Folha: 0
       -> >=:
```

Regras (C45)

```
Sex == female AND SibSp < 6 AND Pclass < 2.5 AND Fare < 28.8562 AND Age < 53.5 AND Parch < 1.
Sex == female AND SibSp < 6 AND Pclass < 2.5 AND Fare < 28.8562 AND Age < 53.5 AND Parch >= 1
Sex == female AND SibSp < 6 AND Pclass < 2.5 AND Fare < 28.8562 AND Age >= 53.5 AND Embarked
Sex == female AND SibSp < 6 AND Pclass < 2.5 AND Fare >= 28.8562 => predict 1
Sex == female AND SibSp < 6 AND Pclass >= 2.5 AND Fare < 32.8813 AND Age < 1.5 => predict 1
Sex == female AND SibSp < 6 AND Pclass >= 2.5 AND Fare < 32.8813 AND Age >= 1.5 AND Embarked
Sex == female AND SibSp < 6 AND Pclass >= 2.5 AND Fare < 32.8813 AND Age >= 1.5 AND Embarked
Sex == female AND SibSp < 6 AND Pclass >= 2.5 AND Fare < 32.8813 AND Age >= 1.5 AND Embarked
Sex == female AND SibSp < 6 AND Pclass >= 2.5 AND Fare >= 32.8813 => predict 0
Sex == female AND SibSp >= 6 => predict 0
Sex == male AND Age < 1.5 => predict 1
Sex == male AND Age >= 1.5 AND Fare < 387.665 AND SibSp < 4.5 AND Parch < 2.5 AND Pclass < 1.
Sex == male AND Age >= 1.5 AND Fare < 387.665 AND SibSp < 4.5 AND Parch < 2.5 AND Pclass >= 1
Sex == male AND Age >= 1.5 AND Fare < 387.665 AND SibSp < 4.5 AND Parch >= 2.5 => predict 0
Sex == male AND Age >= 1.5 AND Fare < 387.665 AND SibSp >= 4.5 => predict 0
Sex == male AND Age >= 1.5 AND Fare >= 387.665 => predict 1
```

CART

Acurácia (treino): 0.8682 \nAcurácia (teste): 0.8258

Matriz de confusão (teste):

labels: 0, 1

||0|1|

|-|-|-|

|0|100|10|

|1|21|47|

Árvore (CART)

```
[Sex · ['female']]
 -> · :
  [Pclass < 2.5]
   -> < :
    [Fare < 28.8562]
     -> < :
      [Fare < 28.2312]
       -> < :
        [Age < 53.5]
         -> < :
          [SibSp < 0.5]
           -> < :
            Folha: 1
           -> >=:
            Folha: 1
         -> >=:
          [Pclass < 1.5]
           -> < :
            Folha: 1
           -> >=:
            Folha: 0
       -> >=:
        Folha: 0
     -> >=:
      Folha: 1
   -> >=:
    [Fare < 20.6625]
     -> < :
      [Age < 7]
       -> < :
        Folha: 1
       -> >=:
        [Fare < 8.0396]
         -> < :
          [Age < 29.25]
           -> < :
            Folha: 1
           -> >=:
            Folha: 0
         -> >=:
          [Fare < 15.8]
           -> < :
            Folha: 0
           -> >=:
            Folha: 1
     -> >=:
      [Parch < 0.5]
       -> < :
        Folha: 1
       -> >=:
        [Age < 5.5]
         -> < :
          [Age < 3.5]
           -> < :
            Folha: 0
           -> >=:
            Folha: 1
         -> >=:
```

Regras (CART)

```
Sex · ['female'] AND Pclass < 2.5 AND Fare < 28.8562 AND Fare < 28.2312 AND Age < 53.5 AND Si
Sex · ['female'] AND Pclass < 2.5 AND Fare < 28.8562 AND Fare < 28.2312 AND Age < 53.5 AND Si
Sex · ['female'] AND Pclass < 2.5 AND Fare < 28.8562 AND Fare < 28.2312 AND Age >= 53.5 AND P
Sex · ['female'] AND Pclass < 2.5 AND Fare < 28.8562 AND Fare < 28.2312 AND Age >= 53.5 AND P
Sex · ['female'] AND Pclass < 2.5 AND Fare < 28.8562 AND Fare >= 28.2312 => predict 0
Sex · ['female'] AND Pclass < 2.5 AND Fare >= 28.8562 => predict 1
Sex · ['female'] AND Pclass >= 2.5 AND Fare < 20.6625 AND Age < 7 => predict 1
Sex · ['female'] AND Pclass >= 2.5 AND Fare < 20.6625 AND Age >= 7 AND Fare < 8.0396 AND Age
Sex · ['female'] AND Pclass >= 2.5 AND Fare < 20.6625 AND Age >= 7 AND Fare < 8.0396 AND Age
Sex · ['female'] AND Pclass >= 2.5 AND Fare < 20.6625 AND Age >= 7 AND Fare >= 8.0396 AND Far
Sex · ['female'] AND Pclass >= 2.5 AND Fare < 20.6625 AND Age >= 7 AND Fare >= 8.0396 AND Far
Sex · ['female'] AND Pclass >= 2.5 AND Fare >= 20.6625 AND Parch < 0.5 => predict 1
Sex · ['female'] AND Pclass >= 2.5 AND Fare >= 20.6625 AND Parch >= 0.5 AND Age < 5.5 AND Age
Sex · ['female'] AND Pclass >= 2.5 AND Fare >= 20.6625 AND Parch >= 0.5 AND Age < 5.5 AND Age
Sex · ['female'] AND Pclass >= 2.5 AND Fare >= 20.6625 AND Parch >= 0.5 AND Age >= 5.5 AND Pa
Sex · ['female'] AND Pclass >= 2.5 AND Fare >= 20.6625 AND Parch >= 0.5 AND Age >= 5.5 AND Pa
Sex · ['female'] AND Age < 6.5 AND SibSp < 2.5 => predict 1
Sex · ['female'] AND Age < 6.5 AND SibSp >= 2.5 AND Parch < 1.5 => predict 0
Sex · ['female'] AND Age < 6.5 AND SibSp >= 2.5 AND Parch >= 1.5 AND Age < 3.5 => predict 1
Sex · ['female'] AND Age < 6.5 AND SibSp >= 2.5 AND Parch >= 1.5 AND Age >= 3.5 => predict 0
Sex · ['female'] AND Age >= 6.5 AND Pclass < 1.5 AND Age < 53 AND Fare < 26.1438 => predict 0
Sex · ['female'] AND Age >= 6.5 AND Pclass < 1.5 AND Age < 53 AND Fare >= 26.1438 AND Fare <
Sex · ['female'] AND Age >= 6.5 AND Pclass < 1.5 AND Age < 53 AND Fare >= 26.1438 AND Fare >=
Sex · ['female'] AND Age >= 6.5 AND Pclass < 1.5 AND Age >= 53 AND SibSp < 0.5 => predict 0
Sex · ['female'] AND Age >= 6.5 AND Pclass < 1.5 AND Age >= 53 AND SibSp >= 0.5 AND Age < 62
Sex · ['female'] AND Age >= 6.5 AND Pclass < 1.5 AND Age >= 53 AND SibSp >= 0.5 AND Age >= 62
Sex · ['female'] AND Age >= 6.5 AND Pclass >= 1.5 AND Fare < 51.6979 AND Embarked · ['C'] AND
Sex · ['female'] AND Age >= 6.5 AND Pclass >= 1.5 AND Fare < 51.6979 AND Embarked · ['C'] AND
Sex · ['female'] AND Age >= 6.5 AND Pclass >= 1.5 AND Fare < 51.6979 AND Embarked · ['C'] AND
Sex · ['female'] AND Age >= 6.5 AND Pclass >= 1.5 AND Fare < 51.6979 AND Embarked · ['C'] AND
Sex · ['female'] AND Age >= 6.5 AND Pclass >= 1.5 AND Fare >= 51.6979 AND Fare < 63.0229 AND
Sex · ['female'] AND Age >= 6.5 AND Pclass >= 1.5 AND Fare >= 51.6979 AND Fare < 63.0229 AND
Sex · ['female'] AND Age >= 6.5 AND Pclass >= 1.5 AND Fare >= 51.6979 AND Fare >= 63.0229 =>
```