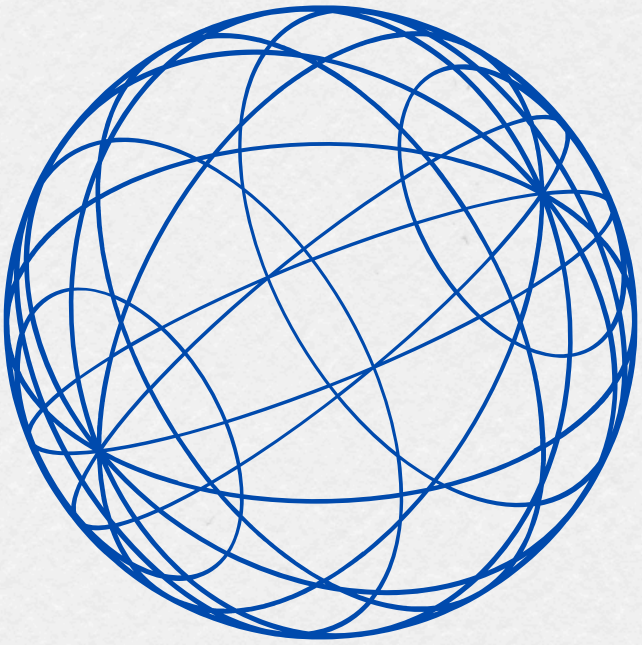
Upen

work of

DECISION TREE AND ML MODELING HOMEWORK

Angel gabriel zuñiga



Presentado por: zuñiga

Heart Disease Dataset Analysis

Dataset Information

Dataset: Heart Disease UCI

Link: https://www.kaggle.com/datasets/cherngs/heart-disease-cleveland-uci

Target variable: target (1 = heart disease, 0 = no heart disease)

Features include: age, sex, cp (chest pain), thalach (max heart rate), chol, exang, etc.

Part 1: Problem Framing

Is this a classification or regression task?

This is a classification task because the target variable is binary: 1 indicates presence of heart disease, 0 indicates absence.

What is the input? What is the output (target)?

The input consists of patient features such as age, sex, cp, chol, thalach, etc.

The output (target) is target, which represents the presence or absence of heart disease.

Is this a supervised or unsupervised learning problem?

This is a supervised learning problem because the dataset includes labeled outcomes (the target variable is known).

Part 2: Feature Understanding

Choose 5 features you think are most relevant. For each feature:

- Is it numerical or categorical?
- Would it need transformation (e.g., scaling, encoding)?
- Could it contain missing values? How would you handle them?

- cp (chest pain type)
  - · Type: Categorical (values 0–3)
  - · Transformation: Encoding required (e.g., one-hot encoding)
  - · Missing values: Check for NaNs. If present, use mode imputation or domain knowledge.
- thalach (maximum heart rate achieved)
  - · Type: Numerical
  - · Transformation: Scaling may help (e.g., StandardScaler)
  - · Missing values: If any, consider imputing with mean/median.
  - · sex (1 = male, 0 = female)
- Type: Categorical (binary)
  - · Transformation: No transformation needed unless doing one-hot encoding.
  - · Missing values: Low chance. If missing, impute with mode.
- chol (serum cholesterol in mg/dl)
  - · Type: Numerical
  - · Transformation: Scaling (e.g., StandardScaler)
  - · Missing values: If any, impute with mean or median.
- exang (exercise-induced angina)
  - · Type: Categorical (binary: 1 = yes, 0 = no)
  - · Transformation: No transformation necessary unless encoding all categorical features.
  - · Missing values: If missing, impute using mode or consider as unknown category.

P**art 3: Decision Tree Modeling Plan (30%)**

Why would a decision tree be appropriate or not for this problem?
A decision tree is appropriate because it can handle both categorical and numerical variables, is easy to interpret, and does not require feature scaling. However, it is prone to overfitting, especially with deep trees.
What hyperparameters would you consider tuning?
- max_depth (maximum tree depth)
 - min_samples_split (minimum samples required to split)
 - min_samples_leaf (minimum samples in a leaf)
 - max_features (number of features to consider at each split)
 - criterion (gini or entropy)
How could you tell if your tree is overfitting?
- Very high accuracy on training data but poor performance on validation/test data
 - Large difference between training and validation scores
 - Very deep tree with many branches

Part 4: Evaluation Strategy
What metrics would you use to evaluate your model and why?
- Accuracy: To measure overall correctness (but may be misleading with imbalanced data).
 - Precision: To see how many predicted positives are actually correct (important if false positives are costly).
 - Recall (Sensitivity): To understand how well the model identifies actual cases of heart disease (important for medical diagnostics).
 - F1 Score: A balance between precision and recall, especially useful if there is class imbalance.
 - ROC-AUC Score: To measure the model's ability to distinguish between classes across thresholds.

Expected Profit Analysis

Expected Profit = (Success Rate × Revenue) - Cost

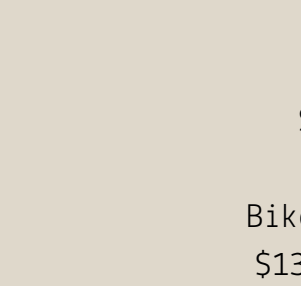
Bike Delivery

- Sunny: (0.9 × 200) - 50 = 180 - 50 = $130
- Rainy: (0.4 × 200) - 50 = 80 - 50 = $30

Car Delivery

- Sunny: (0.7 × 200) - 80 = 140 - 80 = $60
- Rainy: (0.8 × 200) - 80 = 160 - 80 = $80

Decision Tree Representation



Best Strategy Summary

Sunny: Choose Bike Delivery ($130 > $60)

Rainy: Choose Car Delivery ($80 > $30)

If you know the weather forecast:

- Sunny → Bike
- Rainy → Car

If you don’t know the weather and assume a 50/50 chance:

- Expected Profit for Bike = (0.5 × 130) + (0.5 × 30) = $80
- Expected Profit for Car = (0.5 × 60) + (0.5 × 80) = $70

Best overall strategy without forecast: Bike Delivery ($80 > $70)

```
# strategies
strategies = {
"Bike Delivery": {
"Sunny": {"success_rate": 0.9, "cost": 50, "revenue": 200},
"Rainy": {"success_rate": 0.4, "cost": 50, "revenue": 200}
},
"Car Delivery": {
"Sunny": {"success_rate": 0.7, "cost": 80, "revenue": 200},
"Rainy": {"success_rate": 0.8, "cost": 80, "revenue": 200}
}
}

# Function to calculate expected profit
def expected_profit(success_rate, revenue, cost):
return (success_rate * revenue) - cost

# Calculate expected profits
profits = {}
for strategy, weather_data in strategies.items():
profits[strategy] = {}
for weather, values in weather_data.items():
profits[strategy][weather] = expected_profit(
values["success_rate"], values["revenue"], values["cost"]
)

# Print results
for weather in ["Sunny", "Rainy"]:
best_strategy = max(profits, key=lambda s: profits[s][weather])
print(f"Best strategy in {weather}: {best_strategy}
(${profits[best_strategy][weather]:.2f})")

# Assume 50% sunny, 50% rainy
overall_profits = {
strategy: 0.5 * profits[strategy]["Sunny"] + 0.5 * profits[strategy]
["Rainy"]
for strategy in profits
}
best_overall = max(overall_profits, key=overall_profits.get)
print(f"\nBest overall strategy (unknown weather): {best_overall}
(${overall_profits[best_overall]:.2f})")
```