UB Social

CS50AI with Python

8.Shopping

Mateus Schwede

Problemática

- Criar classificador nearest-neighbor;
- Classificador tentará prever se usuário comprará ou não;
- Baseado em dados como: nº de páginas visitadas, se é fim de semana, navegador usado, etc;
- Dados de treino fornecidos em dataset CSV.

Instruções

- Baixe código de https://cdn.cs50.net/ai/2023/x/projects/4/shopping.zip e descompacte-o;
- Execute "pip3 install scikit-learn" para instalar a biblioteca Python scikit-learn.

Funcionamento

- Dataset CSV shopping.csv, possui ~12.000 sessões de usuários, representadas por cada linha;
- Primeiras 6 colunas:
  - Administrative, Informational, ProductRelated: nº de páginas desse tipo visitadas;
  - Administrative_Duration, Informational_Duration, ProductRelated_Duration: tempo gasto nesses tipos de página.
- Outras colunas:
  - BounceRates, ExitRates, PageValues: métricas do Google Analytics;
  - SpecialDay: quão próximo está de data especial (Dia das Mães, Dia dos Namorados, etc.);
  - Month: mês da sessão (abreviado);
  - OperatingSystems, Browser, Region, TrafficType: nº representando essas categorias;
  - VisitorType: "Returning_Visitor" ou outro valor (visitantes novos);
  - Weekend: TRUE ou FALSE (sessão aconteceu no fim de semana).

Funcionamento

- Coluna principal é Revenue, última coluna:
  - Indica se usuário comprou ou não (TRUE ou FALSE);
  - Coluna a ser prevista (label);
  - Todas outras colunas são dados de entrada (evidence).

Funcionamento

- shopping.py contém código principal do projeto;
- Função main:
  - Carrega dados do CSV usando load_data;
  - Divide dados em conjunto de treino e teste;
  - Treina modelo com train_model usando dados de treino;
  - Usa modelo para prever com base nos dados de teste;
  - Avalia modelo com evaluate e imprime resultados (sensibilidade e especificidade).
- Funções load_data, train_model e evaluate precisam ser implementadas.

Especificações

- Função load_data(filename):
  - Receber nome de arquivo CSV como argumento;
  - Deve abrir arquivo e retornar tupla (evidence, labels);
    - evidence: lista com evidências de cada ponto de dados;
    - labels: lista com rótulos de cada ponto de dados.
  - Cada linha da planilha gera evidência e rótulo correspondente;
  - Tamanhos das listas evidence e labels devem ser iguais ao nº de linhas (exceto cabeçalho);
  - Ordem dos dados deve ser preservada (ex: evidence[0] e labels[0] correspondem à 1ª linha);
  - Cada elemento em evidence deve ser lista de 17 valores (todas colunas, exceto última);
  - Valores devem estar na mesma ordem das colunas do arquivo CSV.

Especificações

- Tipos dos dados esperados:
  - int: Administrative, Informational, ProductRelated, Month, OperatingSystems, Browser, Region, TrafficType, VisitorType, Weekend;
  - float: Administrative_Duration, Informational_Duration, ProductRelated_Duration, BounceRates, ExitRates, PageValues, SpecialDay.
- Conversões adicionais:
  - Month: janeiro = 0, fevereiro = 1, ..., dezembro = 11;
  - VisitorType: 1 para visitante que retornou, 0 caso contrário;
  - Weekend: 1 se visita foi no final de semana, 0 caso contrário;
  - labels: 1 se houve compra, 0 caso contrário.
- Exemplo:
  - evidence[0] = [0, 0.0, 0, 0.0, 1, 0.0, 0.2, 0.2, 0.0, 0.0, 1, 1, 1, 1, 1, 1, 0];
  - labels[0] = 0.

Especificações

- Função train_model(evidence, labels):
  - Recebe listas evidence e labels;
  - Retorna classificador k-nearest neighbors (k=1) treinado com esses dados;
  - Utiliza classe KNeighborsClassifier da biblioteca scikit-learn.

Especificações

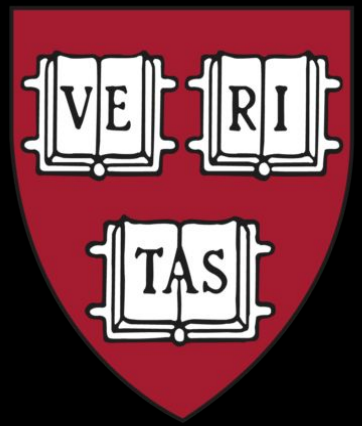
- Função evaluate(labels, predictions):
  - Recebe 2 listas: labels (rótulos reais), predictions (rótulos previstos pelo classificador);
  - Retorna 2 métricas: sensitivity (taxa de verdadeiros positivos (proporção de positivos corretamente previstos)), specificity (taxa de verdadeiros negativos (proporção de negativos corretamente previstos));
  - Rótulos: 1 para usuários que compraram, 0 para que não compraram;
  - Haverá pelo menos 1 positivo e 1 negativo na lista de rótulos reais.

Especificações

- Restrições:
  - Apenas funções especificadas devem ser alteradas no shopping.py;
  - Não modificar shopping.csv.

Submissão

- Visual Studio Code online: https://cs50.dev
- Testar precisão da lógica do algoritmo: check50 ai50/projects/2024/x/shopping
- Testar estilização do código: style50 shopping.py
- Para submissão:
  - Em https://submit.cs50.io/invites/d03c31aef1984c29b5e7b268c3a87b7b, entre com GitHub e autorize CS50;
  - Instale pacote Git, Python 3 (e pip), instalando pacotes: pip3 install style50 check50 submit50
  - Submeta o projeto: submit50 ai50/projects/2024/x/shopping
- Verificar avaliação: https://cs50.me/cs50ai.



UB Social

Mateus Schwede
HBS ID 202400167108 - DCE ID @00963203