Relatório A3 - Estruturas Matemáticas

Nome do professor: Wellington Lacerda | Semestre: 2025.2

Apresentação da Equipe

Alisson Lucas Alves de Oliveira | RA: **1272322139**
Alisson Bezerra Brito | RA: **1272326397**

Situação e Solução Aplicada

Situação: Análise de Tendência do Mercado de Ações

O projeto aborda a **Análise de Tendências do Mercado de Ações**, utilizando um modelo de **Cadeias de Markov**. O objetivo é calcular a probabilidade de o mercado transitar entre três estados possíveis (Alta, Estável, Baixa) ao longo de um determinado número de dias.

Estados Definidos:

1. **Alta (A):** O valor médio das ações subiu.
2. **Estável (E):** O valor médio das ações manteve-se.
3. **Baixa (B):** O valor médio das ações caiu.

A matriz de transição utilizada no exemplo representa as probabilidades de mudança de estado em um dia:

- Alta para Alta: 50%, Estável: 30%, Baixa: 20%
- Estável para Alta: 40%, Estável: 40%, Baixa: 20%
- Baixa para Alta: 30%, Estável: 30%, Baixa: 40%

Solução Aplicada

Foi implementada uma aplicação modular na linguagem **Julia** que realiza:

1. A validação matemática da matriz de transição (garantindo que as linhas somem 1).
2. O cálculo da matriz de probabilidade após **n** dias (passos), utilizando exponenciação de matrizes (P^n).
3. A consulta direta da probabilidade de transição entre dois estados específicos (ex: de "Alta" para "Baixa" em 2 dias).

Instruções de Instalação e Execução

Pré-requisitos

- Ter a linguagem **Julia (v1.x ou superior)** instalada.

Passo a Passo

1. Salve os códigos-fontes (main.jl, UtilModel.jl, MarkovModel.jl) no **mesmo diretório** ou faça o **git clone** do repositório: https://github.com/Alisson-Oliver/a3-markov.
2. Abra o terminal (console) e navegue até a pasta onde os arquivos foram salvos.
3. Execute o comando: **julia main.jl**

Saída Esperada

O sistema exibirá no console a confirmação da validação da matriz e os resultados das probabilidades calculadas para os cenários de teste definidos no arquivo principal.

Códigos-Fontes da Aplicação

Arquivo: src/UtilModel.jl

Responsável pelas funções matemáticas auxiliares e validação.

```julia
module UtilModel

using LinearAlgebra

function validar_matriz(matriz)
    for (i, linha) in enumerate(eachrow(matriz))
        if sum(linha) != 1
            error("Erro: A linha $i ($linha) não soma 1.0 (soma = $(sum(linha))).")
        end
    end
    return true
end

function calcular_matriz_n_passos(matriz, n)
    if n < 1
        error("O número de passos 'n' deve ser 1 ou maior.")
    end
    return matriz^n
end

end
```

Arquivo: src/MarkovModel.jl

Lógica principal do modelo de Markov, gerenciando os estados e chamando as funções utilitárias.

```julia
module MarkovModel

import ..UtilModel

function calcular_probabilidade(matriz, lista_estados, estado_inicial, estado_final, n)
    mapa_estados = Dict()
    for (i, estado) in enumerate(lista_estados)
        mapa_estados[estado] = i
    end

    if !haskey(mapa_estados, estado_inicial)
        error("Estado inicial '$estado_inicial' não encontrado. Estados válidos: $(lista_estados)")
    end

    if !haskey(mapa_estados, estado_final)
        error("Estado final '$estado_final' não encontrado. Estados válidos: $(lista_estados)")
    end

    idx_inicial = mapa_estados[estado_inicial]
    idx_final = mapa_estados[estado_final]

    UtilModel.validar_matriz(matriz)

    matriz_n = UtilModel.calcular_matriz_n_passos(matriz, n)

    return matriz_n[idx_inicial, idx_final]
end

end

```

Arquivo: src/main.jl

Arquivo principal que define os dados do mercado e executa os testes.

```julia
include("UtilModel.jl")
include("MarkovModel.jl")

using .UtilModel
using .MarkovModel

println("--- Resolvedor de Cadeias de Markov (Mercado de Ações) ---")

estados_mercado = ["Alta", "Estável", "Baixa"]

# Colunas: Alta, Estável, Baixa
matriz_mercado = [
    0.5  0.3  0.2;  # Linha "Alta"
    0.4  0.4  0.2;  # Linha "Estável"
    0.3  0.3  0.4   # Linha "Baixa"
]

try
    p1 = MarkovModel.calcular_probabilidade(matriz_mercado,estados_mercado,"Alta","Baixa",1)
    println("Probabilidade (Alta -> Baixa em 1 dia): $(round(p1*100, digits=2))%")

    p2 = MarkovModel.calcular_probabilidade(matriz_mercado,estados_mercado,"Baixa","Alta",1)
    println("Probabilidade (Baixa -> Alta em 1 dia): $(round(p2*100, digits=2))%")

    p3 = MarkovModel.calcular_probabilidade(matriz_mercado,estados_mercado,"Alta","Baixa",2)
    println("Probabilidade (Alta -> Baixa em 2 dias): $(round(p3*100, digits=2))%")

    p4 = MarkovModel.calcular_probabilidade(matriz_mercado,estados_mercado,"Estável","Alta",3)
    println("Probabilidade (Estável -> Alta em 3 dias): $(round(p4*100, digits=2))%")
catch e
    println("Erro durante a execução:")
    println(e)
end
```