

UNIVERSIDADE PRESBITERIANA MACKENZIE

Integrantes:

Bruno Zovaro Nascimento - 10424880

Douglas Novaes Dias – 14023666

Milan Mirco Moraes Mazur – 10363757

Paulo Cesar Masson Junior - 10416023

PROJETO APLICADO II

SUBCONJUNTO DE PREVISÃO DE IDADE

São Paulo

2024

SUMÁRIO – Aplicando Conhecimento – Aula 2

- 1 Definição da linguagem de programação usada no projeto
- 2 Análise exploratória da base de dados escolhida
- 3 Tratamento da base de dados (Preparação e treinamento)
- 4 Definição e descrição das bases teóricas dos métodos
- 5 Definição e descrição de como será calculada a acurácia
- 6 Base de Dados/ Repositório do Github
- 7 Cronograma de Atividades

1. Definição da linguagem de programação usada no projeto

A linguagem de programação é Python. As bibliotecas a serem usadas incluem pandas, seaborn, fetch_ucirepo, sklearn.

```
Importando as bibliotecas

✓ from ucimlrepo import fetch_ucirepo
  import pandas as pd
  import seaborn as sns
  from IPython.display import display

✓ 9.2s
```

2. Análise exploratória da base de dados escolhida.

Head:

```
display(dataBase.head(10))
```

✓ 0.0s

	SEQN	age_group	RIDAGEYR	RIAGENDR	PAQ605	BMXBMI	LBXGLU	DIQ010	LBXGLT	LBXIN
0	73564.0	Adult	61.0	2.0	2.0	35.7	110.0	2.0	150.0	14.91
1	73568.0	Adult	26.0	2.0	2.0	20.3	89.0	2.0	80.0	3.85
2	73576.0	Adult	16.0	1.0	2.0	23.2	89.0	2.0	68.0	6.14
3	73577.0	Adult	32.0	1.0	2.0	28.9	104.0	2.0	84.0	16.15
4	73580.0	Adult	38.0	2.0	1.0	35.9	103.0	2.0	81.0	10.92
5	73581.0	Adult	50.0	1.0	2.0	23.6	110.0	2.0	100.0	6.08
6	73587.0	Adult	14.0	1.0	2.0	38.7	94.0	2.0	202.0	21.11
7	73596.0	Adult	57.0	2.0	2.0	38.3	107.0	2.0	164.0	20.93
8	73607.0	Senior	75.0	1.0	2.0	38.9	89.0	2.0	113.0	17.47
9	73610.0	Adult	43.0	1.0	1.0	28.9	90.0	2.0	95.0	3.24

Describe:

```
display(dataBase.describe())
```

✓ 0.0s

	sk	idade	imc	nivelGlicose	LBXGLT	nivelInsulinaSangue
count	2278.000000	2278.000000	2278.000000	2278.000000	2278.000000	2278.000000
mean	78691.853819	41.795874	27.955180	99.553117	114.978929	11.834794
std	2921.365151	20.156111	7.248962	17.889834	47.061239	9.718812
min	73564.000000	12.000000	14.500000	63.000000	40.000000	0.140000
25%	76171.750000	24.000000	22.800000	91.000000	87.000000	5.860000
50%	78749.000000	41.000000	26.800000	97.000000	105.000000	9.040000
75%	81213.500000	58.000000	31.200000	104.000000	130.000000	14.440000
max	83727.000000	80.000000	70.100000	405.000000	604.000000	102.290000

3. Tratamento da base de dados (Preparação e treinamento).

Preparação 1:

```
Tratamento da base

dataBase = dataBase.rename(columns={
    'SEQN': 'sk',
    'age_group': 'faixaEtaria',
    'RIDAGEYR': 'idade',
    'RIAGENDR': 'sexo',
    'PAQ605': 'exerciciosSemanais?',
    'BMXBMI': 'imc',
    'LBXGLU': 'nivelGlicose',
    'DIQ010': 'diabetico?',
    'LBXGLT': 'LBXGLT',
    'LBXIN': 'nivelInsulinaSangue'
})

dataBase["sexo"] = dataBase["sexo"].replace({1: 'MALE', 2: 'FEMALE'})
dataBase["exerciciosSemanais?"] = dataBase["exerciciosSemanais?"].replace({1: 'SIM', 2: 'NAO'})
dataBase["diabetico?"] = dataBase["diabetico?"].replace({1: 'SIM', 2: 'NAO'})
```

Resultado Preparação 1:

	sk	faixaEtaria	idade	sexo	exerciciosSemanais	imc	nivelGlicose	diabetico	LBXGLT	nivelInsulinaSangue
0	73564.0	Adult	61.0	FEMALE	NAO	35.7	110.0	NAO	150.0	14.91
1	73568.0	Adult	26.0	FEMALE	NAO	20.3	89.0	NAO	80.0	3.85
2	73576.0	Adult	16.0	MALE	NAO	23.2	89.0	NAO	68.0	6.14
3	73577.0	Adult	32.0	MALE	NAO	28.9	104.0	NAO	84.0	16.15
4	73580.0	Adult	38.0	FEMALE	SIM	35.9	103.0	NAO	81.0	10.92
5	73581.0	Adult	50.0	MALE	NAO	23.6	110.0	NAO	100.0	6.08
6	73587.0	Adult	14.0	MALE	NAO	38.7	94.0	NAO	202.0	21.11
7	73596.0	Adult	57.0	FEMALE	NAO	38.3	107.0	NAO	164.0	20.93
8	73607.0	Senior	75.0	MALE	NAO	38.9	89.0	NAO	113.0	17.47
9	73610.0	Adult	43.0	MALE	SIM	28.9	90.0	NAO	95.0	3.24

Treinamento:

Treinamento de teste

```
x = DataBase['idade']  
y = DataBase[['imc', 'nivelGlicose']]
```

✓ 0.0s

```
if x.isnull().any() or y.isnull().any().any(): # Verifica se há valores ausentes em y  
    print("Valores ausentes encontrados. Removendo linhas com valores ausentes.")  
    DataBase = DataBase.dropna(subset=['idade', 'imc', 'nivelGlicose']) # Remover linhas com valores ausentes  
x = DataBase['idade']  
y = DataBase[['imc', 'nivelGlicose']]
```

✓ 0.0s

```
x_train, x_test, y_train, y_test = train_test_split(  
    x, y,  
    test_size=0.3,  
    random_state=123  
)
```

✓ 0.0s

```
print("Dimensões do conjunto de treino:")  
print(f"x_train: {x_train.shape}, y_train: {y_train.shape}")  
print("Dimensões do conjunto de teste:")  
print(f"x_test: {x_test.shape}, y_test: {y_test.shape}")
```

✓ 0.0s

Dimensões do conjunto de treino:
x_train: (1594,), y_train: (1594, 2)
Dimensões do conjunto de teste:
x_test: (684,), y_test: (684, 2)

```
x_train, x_test, y_train, y_test = train_test_split(  
    x, y,  
    test_size=0.3,  
    random_state=123  
)
```

```
# Criação da saída multi regressão  
regressor = MultiOutputRegressor(LinearRegression())
```

```
# Ajuste do modelo  
regressor.fit(x_train.values.reshape(-1, 1), y_train)
```

```
# Previsões  
y_pred = regressor.predict(x_test.values.reshape(-1, 1))
```

```
# Avaliação do desempenho para cada saída  
for i, coluna in enumerate(y_train.columns):  
    mse = mean_squared_error(y_test[coluna], y_pred[:, i])  
    print(f"Erro quadrático médio para {coluna}: {mse}")
```

✓ 0.0s

Erro quadrático médio para imc: 50.378884502155785
Erro quadrático médio para nivelGlicose: 402.8259095253492

4. Definição e descrição das bases teóricas dos métodos.

O método a ser usado, inicialmente definido, é a Regressão Logística.

5. Definição e descrição de como será calculada a acurácia

A acurácia mede a eficiência de um modelo. No nosso escopo de trabalho, usaremos a acurácia para calcular quanto de sucesso obtivemos (em porcentagem) em prever a idade dos entrevistados. Também poderemos ver qual das medidas nos fornece uma melhor eficiência, se são as medidas fisiológicas ou bioquímicas.

Para isso, utilizaremos uma faixa de ± 1 . Por exemplo, caso a idade de uma pessoa seja 40 e o nosso modelo prever entre 39 e 41, podemos considerar como correto. Se tiver fora dessa faixa, será uma previsão incorreta.

Com isso, pegamos os valores preditos corretamente e dividimos pelo total de casos no modelo e obtemos a acurácia. Há algumas formas de fazermos isso no Python.

$$\text{Acurácia} = \frac{\text{Casos Preditos corretamente}}{\text{Total de casos}}$$

6. Base de Dados/ Repositório do Github

Links das bases de dados:

[https://archive.ics.uci.edu/dataset/887/
national+health+and+nutrition+health+survey+2013-2014+\(nhanes\)
+age+prediction+subset](https://archive.ics.uci.edu/dataset/887/national+health+and+nutrition+health+survey+2013-2014+(nhanes)+age+prediction+subset)

Link para acesso ao GitHub:

<https://github.com/pcmassonjr/ProjetoAplicado2>

7. Cronograma de Atividades

cronograma do projeto

PROJETO Nº 2

PROJETO APLICADO 2

14/09/2024

Definição do grupo;
Premissas do projeto;
Objetivos e metas;

05/10/2024

Definição das bibliotecas;
Análise exploratória dos dados
Tratamento das bases;
Definição bases teóricas dos métodos e acurácia

02/11/2024

Consolidação do método analítico;
Aplicação das medias de acurácia;
Definição dos resultados preliminares e esboço do storytelling;

23/11/2024

Relatório técnico do projeto;
Apresentação do storytelling e video;
Disponibilização do repositório do Github;

Finalizado