Relatório de mineração de dados - Pré-processamento

Silas Leme Silvério – BI1760343

O objetivo das técnicas de pré-processamento de dados é preparar os dados brutos para serem analisados.

Informações de Data Set

O data set escolhido se chama HCC Survival Data Set possui 49 colunas, contendo 165 instancias no total. A característica de seus atributos é de números inteiros e reais, associando os dados de modo a classificá-los.

O conjunto de dados de HCC foi obtido em um Hospital Universitário em Portugal e contém vários dados demográficos, fatores de risco, dados laboratoriais e de sobrevida global de 165 pacientes reais com diagnóstico de HCC. O conjunto de dados contém 49 recursos selecionados de acordo com as Diretrizes de Prática Clínica da EASL-EORTC (Associação Europeia para o Estudo do Fígado - Organização Europeia para Pesquisa e Tratamento do Câncer), que são o estado da arte atual sobre o gerenciamento de HCC.

Trata-se de um conjunto de dados heterogêneo, com 23 variáveis quantitativas e 26 variáveis qualitativas. No geral, os dados ausentes representam 10,22% de todo o conjunto de dados e apenas oito pacientes têm informações completas em todos os campos (4,85%). A variável alvo é a sobrevivência em 1 ano, e foi codificada como uma variável binária: 0 (morre) e 1 (vive). Um certo grau de desequilíbrio de classes também está presente

Informações das colunas

Gender: nominal
Symptoms: nominal
Alcohol: nominal
Hepatitis B Surface Antigen: nominal
Hepatitis B e Antigen: nominal
Hepatitis B Core Antibody: nominal
Hepatitis C Virus Antibody: nominal
Cirrhosis : nominal
Endemic Countries: nominal
Smoking: nominal
Diabetes: nominal
Obesity: nominal
Hemochromatosis: nominal
Arterial Hypertension: nominal
Chronic Renal Insufficiency: nominal
Human Immunodeficiency Virus: nominal
Nonalcoholic Steatohepatitis: nominal

Esophageal Varices: nominal
Splenomegaly: nominal
Portal Hypertension: nominal
Portal Vein Thrombosis: nominal
Liver Metastasis: nominal
Radiological Hallmark: nominal
Age at diagnosis: integer
Grams of Alcohol per day: continuous
Packs of cigarets per year: continuous
Performance Status: ordinal
Encefalopathy degree: ordinal
Ascites degree: ordinal
International Normalised Ratio: continuous
Alpha-Fetoprotein (ng/mL): continuous
Haemoglobin (g/dL): continuous
Mean Corpuscular Volume (fl): continuous
Leukocytes(G/L): continuous
Platelets (G/L): continuous
Albumin (mg/dL): continuous
Total Bilirubin(mg/dL): continuous
Alanine transaminase (U/L): continuous
Aspartate transaminase (U/L): continuous
Gamma glutamyl transferase (U/L): continuous
Alkaline phosphatase (U/L): continuous
Total Proteins (g/dL): continuous
Creatinine (mg/dL): continuous
Number of Nodules: integer
Major dimension of nodule (cm): continuous
Direct Bilirubin (mg/dL): continuous
Iron (mcg/dL): continuous
Oxygen Saturation (%): continuous
Ferritin (ng/mL): continuous
Class: nominal (1 se o paciente sobreviveu, 0 se o paciente morreu)

Informações do Código

Data Cleaning

Para essa etapa do pré-processamento, constituída da limpeza dos dados foi utilizado como base o código disponibilizado pelo professor.

Algumas alterações foram feitas para que o algoritmo realizasse o tratamento de forma correta:

- Especificação do caminho para acesso ao arquivo com os dados.

```
input_file = '0-Datasets/hcc-data.txt'
```

- Input do nome de todas as colunas do Data Set.

```
df = pd.read_csv(input_file, # Nome do arquivo com dados
                 names =['Gender', 'Symptoms', 'Alcohol', 'Hepatitis B Surface Antigen', 'Hepatitis B
                 e Antigen', 'Hepatitis B Core Antibody', 'Hepatitis C Virus Antibody', 'Cirrhosis',
                 'Endemic Countries', 'Smoking', 'Diabetes', 'Obesity', 'Hemochromatosis', 'Arterial
                 Hypertension', 'Chronic Renal Insufficiency', 'Human Immunodeficiency Virus',
                 'Nonalcoholic Steatohepatitis', 'Esophageal Varices', 'Splenomegaly', 'Portal
                 Hypertension', 'Portal Vein Thrombosis', 'Liver Metastasis', 'Radiological Hallmark',
                 'Age at diagnosis', 'Grams of Alcohol per day', 'Packs of cigarets per year',
                 'Performance Status', 'Encefalopathy degree', 'Ascites degree', 'International
                 Normalised Ratio', 'Alpha-Fetoprotein (ng/mL)', 'Haemoglobin (g/dL)', 'Mean
                 Corpuscular Volume (fl)', 'Leukocytes(G/L)', 'Platelets (G/L)', 'Albumin (mg/dL)',
                 'Total Bilirubin(mg/dL)', 'Alanine transaminase (U/L)', 'Aspartate transaminase (U/L)
                 ', 'Gamma glutamyl transferase (U/L)', 'Alkaline phosphatase (U/L)', 'Total Proteins
                 (g/dL)', 'Creatinine (mg/dL)', 'Number of Nodules', 'Major dimension of nodule (cm)',
                 'Direct Bilirubin (mg/dL)', 'Iron (mcg/dL)', 'Oxygen Saturation (%)', 'Ferritin (ng/
                 mL)','Class'], # Nome das colunas
```

- Seleção das tabelas que seriam utilizadas para o processo de limpeza.

```
usecols = ['Grams of Alcohol per day', 'Packs of cigarets per year', 'International
Normalised Ratio', 'Alpha-Fetoprotein (ng/mL)', 'Haemoglobin (g/dL)', 'Mean
Corpuscular Volume (fl)', 'Leukocytes(G/L)', 'Platelets (G/L)', 'Albumin (mg/dL)',
'Total Bilirubin(mg/dL)', 'Alanine transaminase (U/L)', 'Aspartate transaminase (U/L)
', 'Gamma glutamyl transferase (U/L)', 'Alkaline phosphatase (U/L)', 'Total Proteins
(g/dL)', 'Creatinine (mg/dL)', 'Number of Nodules', 'Major dimension of nodule (cm)',
'Direct Bilirubin (mg/dL)', 'Iron (mcg/dL)', 'Oxygen Saturation (%)', 'Ferritin (ng/
mL)'], # Define as colunas que serão  utilizadas
```

Foram selecionadas essas colunas devido ao fato de haver nelas maior quantidade de valores ausentes, além de que muitas colunas possuírem valores nominais (1 - sim e 0 - não). Essas não foram consideradas para a limpeza, apenas as colunas com valores contínuos.

- Definição do caractere que sinaliza um valor ausente no arquivo com os dados.

```
na_values='?') # Define que ? será considerado valores ausentes
```

Como descrito acima essa data set possui diversos valores faltantes (10,22%), definidos com “?” no arquivo com os dados.

Para a configuração dos arquivos foram selecionados como arquivo de entrada o Data set original obtido no repositório de Data Sets e como arquivo de saída foi criado documento de texto chamado hcc-dataClear onde foram armazenados os dados depois do pré-processamento com as colunas preenchidas e sem dados faltantes.

```
output_file = '0-Datasets/hcc-dataClear.txt'
input_file = '0-Datasets/hcc-data.txt'
```

Algumas das funções iniciais não foram alteradas. Essas são funções secundárias, mas que auxiliam na visualização e estudo dos dados.

```
# Imprime as 15 primeiras linhas do arquivo
print("PRIMEIRAS 15 LINHAS\n")
print(df.head(15))
print("\n")

# Imprime informações sobre dos dados
print("INFORMAÇÕES GERAIS DOS DADOS\n")
print(df.info())
print("\n")

# Imprime uma analise descritiva sobre dos dados
print("DESCRIÇÃO DOS DADOS\n")
print(df.describe())
print("\n")

# Imprime a quantidade de valores faltantes por coluna
print("VALORES FALTANTES\n")
print(df.isnull().sum())
print("\n")
```

```
VALORES FALTANTES

Grams of Alcohol per day              48
Packs of cigarets per year            53
International Normalised Ratio         4
Alpha-Fetoprotein (ng/mL)               8
Haemoglobin (g/dL)                      3
Mean Corpuscular Volume (fl)            3
Leukocytes(G/L)                         3
Platelets (G/L)                         3
Albumin (mg/dL)                         6
Total Bilirubin(mg/dL)                  5
Alanine transaminase (U/L)              4
Aspartate transaminase (U/L)            3
Gamma glutamyl transferase (U/L)        3
Alkaline phosphatase (U/L)              3
Total Proteins (g/dL)                  11
Creatinine (mg/dL)                      7
Number of Nodules                       2
Major dimension of nodule (cm)         20
Direct Bilirubin (mg/dL)               44
Iron (mcg/dL)                          79
Oxygen Saturation (%)                  80
Ferritin (ng/mL)                       80
dtype: int64
```

A função principal dessa etapa de pré-processamento é chamada de "*UpdateMissingvalue"*. Essa função é chamada dentro de um laço de repetição "for" que percorre cada coluna do Data set onde existem valores ausentes armazenados em uma variável chamada de "*columns_missing_values*".

```python
columns_missing_value = df.columns[df.isnull().any()]
print(columns_missing_value)
method = 'mode' # number or median or mean or mode

for c in columns_missing_value:
    UptateMissingvalue(df, c)

print('Total valores ausentes: ' + str(df['Symptoms'].isnull().sum()))
print(df.describe())
print("\n")
print(df.head(15))
print(df_original.head(15))
print("\n")
```

Na função UpdateMissingvalues são passados como parametro o DataFrame (df) que contem os dados, o indice da coluna (column) onde existem valores faltantes, o método (number, median, mean ou mode) que será utilizado para realozação do preenhimento dessas lacunas e o numero que será usado csa o método number seja escolhido.

Dentro da função está uma sequencia de laços condicionais que verificam qual a opção de método passada como parametro para que assim realize uma determinada ação conforme a escolha.

```python
def UptateMissingvalue(df, column, method="mode", number=0):
    if method == 'number':
        # Substituindo valores ausentes por um número
        df[column].fillna(number, inplace=True)
    elif method == 'median':
        # Substituindo valores ausentes pela mediana
        median = df['Symptoms'].median()
        df[column].fillna(median, inplace=True)
    elif method == 'mean':
        # Substituindo valores ausentes pela média
        mean = df[column].mean()
        df[column].fillna(mean, inplace=True)
    elif method == 'mode':
        # Substituindo valores ausentes pela moda
        mode = df[column].mode()[0]
        df[column].fillna(mode, inplace=True)
```

O método utilizado nessa atividade foi o de moda, pois diversas colunas possuem valores discretos definidos como 1 ou 0 por isso seria interessante utilizar aqueles que mais se repetem dada as certas condições, além de se tratar de dados sobre uma doença.

Data Normalization

Para realização dessa etapa de pré-processamento foi utilizado como arquivo de entrada o documento gerado como saida da etapa anterior de limpeza dos dados, (hcc-dataClear.txt)

```python
input_file = '0-Datasets/hcc-dataClear.txt'
```

Da mesma maneira que na etapa anterios foram informados os nomes das colunas (names), bem como aquelas que seriam utilizadas na normalização (features).

Uma diferença realizada nesse código foi a identificação do *target*, isso é, aquela coluna que representa uma “resposta” ou que na analize das demais colunas nos da uma informação, no caso do Data set apresentado a coluna *Class* representa o target, pois informa se o paciente sobreviveu ou se ele morreu através dos valores 1 e 0, respectivamente.

Após isso foi realizada a separação dos valores das *features* e do *target*, armazenando esses dados nas variéveis x e y, respectivamente.

```
# Separating out the features
x = df.loc[:, features].values

# Separating out the target
y = df.loc[:,[target]].values
```

Foi realizada então a normalização Z (Z-Score) também conhecida como desvio padrão, que nos permite dizer o quão distante os valores estão da média. Em termos mais tecnicos é a medida de quantos desvios acima ou abaixo de uma pupulação média um dado cru está.

```
# Z-score normalization
x_zcore = StandardScaler().fit_transform(x)
normalized1Df = pd.DataFrame(data = x_zcore, columns = features)
normalized1Df = pd.concat([normalized1Df, df[[target]]], axis = 1)
ShowInformationDataFrame(normalized1Df,"Dataframe Z-Score Normalized")
```

Além da *Z-score normalization* também foi realizada a normalização de máximos e mínimos (*Min-Max Normalization*) que é uma das mais comuns. Consiste em definir, para cada uma das *feature*, um valor mínimo que é transformado em 0 e o valor máximo que é transformado em 1, dessa forma todos os outros valores são transformados em valores decimais entre 1 e 0.

```
# Mix-Max normalization
x_minmax = MinMaxScaler().fit_transform(x)
normalized2Df = pd.DataFrame(data = x_minmax, columns = features)
normalized2Df = pd.concat([normalized2Df, df[[target]]], axis = 1)
ShowInformationDataFrame(normalized2Df,"Dataframe Min-Max Normalized")
```

É possível realizar a comparação dos dados através da visualização dos dados do *DataFrame* original.

```
def ShowInformationDataFrame(df, message=""):
    print(message+"\n")
    print(df.info())
    print(df.describe())
    print(df.head(10))
    print("\n")
```

Data Reduction

Para realização dessa etapa de pré-processamento foi utilizado como arquivo de entrada o documento gerado como saida da etapa anterior de limpeza dos dados, (hcc-dataClear.txt)

```
input_file = '0-Datasets/hcc-dataClear.txt'
```

Da mesma maneira que nas etapas anteriores foram informados os nomes das colunas (names), bem como aquelas que seriam utilizadas na normalização (features), target e a separação desses dados atraves dos comandos

```
# Separating out the features
x = df.loc[:, features].values

# Separating out the target
y = df.loc[:, [target]].values
```

Após isso os as colunas foram padronizadas através do bloco de comandos:

```
# Standardizing the features
x = StandardScaler().fit_transform(x)
normalizedDf = pd.DataFrame(data=x, columns=features)
normalizedDf = pd.concat([normalizedDf, df[[target]]], axis=1)
ShowInformationDataFrame(normalizedDf, "Dataframe Normalized")
```

E foi realizada a projeção PCA (Principal Component Analysis) que é responsável por reduzir a dimensão dos dados, ou seja, um DataFrame com n número de colunas pode ser projetado em um subespaço de um numero menor de colunas, mantendo a essência dos dados.

```
# PCA projection
pca = PCA()
principalComponents = pca.fit_transform(x)
print("Explained variance per component:")
print(pca.explained_variance_ratio_.tolist())
print("\n\n")

principalDf = pd.DataFrame(data=principalComponents[:, 0:2],
                           columns=['principal component 1',
                                    'principal component 2'])
finalDf = pd.concat([principalDf, df[[target]]], axis=1)
ShowInformationDataFrame(finalDf, "Dataframe PCA")

VisualizePcaProjection(finalDf, target)
```

Além disso os dados também foram plotados, através da função VisualizePcaProjection, para uma melhor análise e avaliação dos dados.

```python
def VisualizePcaProjection(finalDf, targetColumn):
    fig = plt.figure(figsize=(8, 8))
    ax = fig.add_subplot(1, 1, 1)
    ax.set_xlabel('Principal Component 1', fontsize=15)
    ax.set_ylabel('Principal Component 2', fontsize=15)
    ax.set_title('2 component PCA', fontsize=20)
    targets = [0, 1, ]
    colors = ['r', 'g']
    for target, color in zip(targets, colors):
        indicesToKeep = finalDf[targetColumn] == target
        ax.scatter(finalDf.loc[indicesToKeep, 'principal component 1'],
                   finalDf.loc[indicesToKeep, 'principal component 2'],
                   c=color, s=50)
    ax.legend(targets)
    ax.grid()
    plt.show()
```