Universidade Presbiteriana Mackenzie

FILA DE ESPERA PARA TRANSPLANTE RENAL NO BRASIL

Eduardo Nogueira Mota

Enzo Vemado

Jenifer Rinara

Vagner Milani

Trabalho apresentado como critério de avaliação da disciplina

PROJETO APLICADO II (Turma 03A).

Professor : Anderson Adaime de Borba

São Paulo

2024

Universidade Presbiteriana Mackenzie

SUMÁRIO

1. MÉTODO ANALÍTICO

1.1. Preparação e Exploração dos Dados

1.1.1. Carregamento dos Dados

Os dados foram carregados a partir de um arquivo CSV contendo informações demográficas e médicas dos pacientes na fila de espera por um transplante de rim. Utilizamos a biblioteca Pandas para leitura e manipulação dos dados.

Código utilizado:

```
file_path = 'waitlist_kidney_brazil.csv'
data = pd.read_csv(file_path, encoding='ISO-8859-1')
```

1.1.2. Limpeza e Tratamento de Dados

Os dados foram limpos tratando valores ausentes e removendo registros onde os pacientes foram removidos da lista de espera para garantir a precisão das previsões. Variáveis categóricas foram transformadas em variáveis numéricas utilizando a codificação *dummy*

Código utilizado:

```
from sklearn.preprocessing import LabelEncoder

# criar uma copia do dataframe para manipulacoes
df = data.copy()

# codificar variaveis categoricas
label_encoders = {}
categorical_columns = ['sex', 'race', 'Blood_type', 'underline_disease', 'age_cat']

for col in categorical_columns:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le

# verificar se ha valores ausentes nas colunas relevantes e preenche-los se necessario
missing_values = df.isnull().sum()

# preenchendo valores ausentes na coluna 'time_on_Dialysis' com a media, ja que e uma variavel crucial
df['time_on_Dialysis'].fillna(df['time_on_Dialysis'].mean(), inplace=True)

# mostrar a nova estrutura do dataframe
df.head(), missing_values[missing_values > 0]
```

1.2. Seleção e Redução de Variáveis

1.2.1. Seleção Inicial de Variáveis

Selecionamos variáveis iniciais com base na relevância clínica e demográfica, como idade, sexo, raça, tempo em diálise e tipo sanguíneo.

1.2.2. Análise de Redução de Dimensionalidade com LASSO

Utilizamos a Regressão Lasso para identificar e reter as variáveis mais significativas, penalizando e potencialmente reduzindo a zero os coeficientes de variáveis menos informativas.

Código utilizado:

```
from sklearn.linear_model import Lasso
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# selecionar variaveis que parecem ser relevantes
features = ['Time_Tx','age_at_list_registration', 'age_cat', 'time_on_Dialysis', 'race', 'sex', 'diabetes', 'Blood_type']
target = 'time'  # tempo de espera na fila como variável alvo

# preparando dados para o modelo
X = df[features]
y = df[target]

# dividir dados em conjuntos de treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)

# escalar os dados
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# ajustar o modelo Lasso
lasso = Lasso(alpha=0.1)
lasso.fit(X_train_scaled, y_train)

# exibir os coeficientes do modelo
feature_importance = pd.DataFrame({'Feature': features, 'Coefficient': lasso.coef_})
feature_importance
```

Após a primeira análise dos coeficientes, os quais foram utilizadas apenas as variáveis numéricas, refizemos a analise utilizando as variáveis categóricas em conjunto.

Código utilizado:

```
# preparando dados para o modelo com todas as colunas numericas disponiveis
all_features = df.select_dtypes(include=[np.number]).columns.tolist()
all_features.remove('time')  # removendo a variavel alvo dos preditores
# all_features.remove('Time_Tx')
X_all = df[all_features]
y_all = df['time']

# dividir dados em conjuntos de treino e teste para todas as colunas
X_train_all, X_test_all, y_train_all, y_test_all = train_test_split(X_all, y_all, test_size=0.3, random_state=0)

# escalar os dados para todas as colunas
scaler_all = StandardScaler()
X_train_all_scaled = scaler_all.fit_transform(X_train_all)
X_test_all_scaled = scaler_all.transform(X_test_all)

# ajustar o modelo Lasso com todas as colunas
lasso_all = Lasso(alpha=0.1)
# preenchendo valores ausentes
x_all_transformada = X_all.fillna(X_all.mean())

# escalar os dados preenchidos
x_all_transformada_treino = scaler_all.fit_transform(x_all_transformada)

# ajustar o modelo Lasso
lasso_all.fit(x_all_transformada_treino, y_all)

# exibir os coeficientes para ver a importancia das colunas
feature_importance_all_filled = pd.DataFrame({'Feature': all_features, 'Coefficient': lasso_all.coef_})
important_features = feature_importance_all_filled[feature_importance_all_filled['Coefficient'] != 0].sort_values(by='Coefficient', key=abs, ascending=False)
important_features
```

Com isso obtivemos os seguintes coeficientes de correlação:

Atributo	Coeficiente
Time_Tx	911.7877
Time_death	19.04528
X36MthsTx	-18.157
calculated_frequency_DR.f	-7.02883
calculated_frequency_DR.f2	6.354734
age_cat	-5.17432
calculated_frequency_A.f	-4.18442
calculated_frequency_A.f2	3.808732
calculated_frequency_DR.f1	3.378271
number_transfusion	-3.19006
HLA_A1	-2.5807
number_gestation	-2.32019
time_on_Dialysis	-2.23765
HLA_A2	2.21863
HLA_DR1	-2.12151
event	-2.10179
calculated_frequency_B.f1	-2.03607
age_at_list_registration	-1.98327
calculated_frequency_B.f2	1.449928

number_prior_transplant	-0.97604
HLA_DR2	0.913288
race	0.611684
underline_disease	-0.59824
HLA_B2	-0.59029
diabetes	-0.42922
cPRA	0.349848
Id	0.331439
Blood_type	-0.2314
HLA_B1	-0.07525
calculated_frequency_A.f1	0.058826
calculated_frequency_B.f	0.050095

1.3. Modelagem Estatística

1.3.1. Construção do Modelo de Regressão Logística:

Com as variáveis selecionadas, construímos um modelo de Regressão Logística para prever se o tempo de espera é maior ou menor que a mediana.

Código utilizado:

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# selecionar apenas as features com maior coefieciente
selected_features = [ 'Time_Tx','Time_death', 'X36MthsTx', 'calculated_frequency_DR.f', 'calculated_frequency_DR.f2', 'age_cat']

x_numericas = x_all_transformada[selected_features]

# get dummies para colunas categoricas
x_colunas = pd.concat([x_numericas, df_dummies], axis=1)
x_colunas = pd.concat([x_colunas, y_all], axis=1)

x_colunas= x_colunas[x_colunas['removed_list_Sim']==True]
y_all= x_colunas['time']
y_selected = (y_all > y_all.median()).astype(int)  # variavel binaria baseada na mediana do tempo de espera
x_colunas = x_colunas.drop('time', axis=1)

log_reg_enh = LogisticRegression(max_iter=1000)
```

1.3.2. Treinamento e Teste do Modelo:

Dividimos os dados em conjuntos de treinamento e teste para validar a eficácia do modelo, treinando-o com o conjunto de treinamento e testando-o posteriormente para verificar sua precisão.

Código utilizado:

```
# treino e teste
X_train_col, X_test_col, y_train_sel, y_test_sel = train_test_split(x_colunas, y_selected, test_size=0.3, random_state=0)

# escalar os dados
X_train_col_scaled = scaler.fit_transform(X_train_col)
X_test_col_scaled = scaler.transform(X_test_col)

#modelo com todas as variaveis
log_reg_enh.fit(X_train_col_scaled, y_train_sel)

y_pred_enh = log_reg_enh.predict(X_test_col_scaled)
```

2. CÁLCULO DAS MÉTRICAS

Após o treinamento, aplicamos o modelo ao conjunto de teste para prever se os pacientes teriam um tempo de espera acima da mediana. As seguintes métricas de desempenho foram calculadas:

- Acurácia: Proporção de previsões corretas feitas pelo modelo.
- Precisão: Proporção de identificações positivas corretas.
- Recall: Proporção de positivos reais corretamente identificados
- F1-Score: Combinação de precisão e recall.

Código utilizado:

```
    # metricas de desempenho
    accuracy_enh = accuracy_score(y_test_sel, y_pred_enh)
    precision_sel = precision_score(y_test_sel, y_pred_enh)
    recall_enh = recall_score(y_test_sel, y_pred_enh)
    f1_enh = f1_score(y_test_sel, y_pred_enh)

    print(f"Acuracia: {accuracy_enh}\nPrecisao: {precision_sel}\nRecall: {recall_enh}\nF1: {f1_enh} ")
```

✓ 0.0s

```
Acuracia: 0.9827849204619743
Precisao: 0.9763676148796498
Recall: 0.9889184397163121
F1: 0.982602950891874
```

3. DESCRIÇÃO DOS RESULTADOS PRELIMINARES

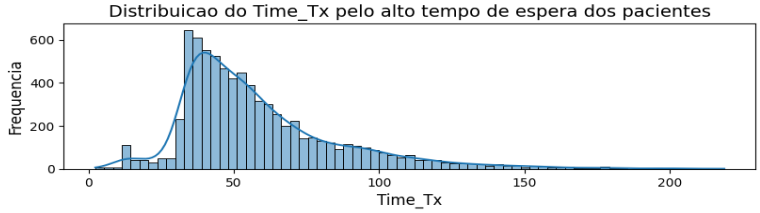
3.1. Análise Estatística e Identificação de Padrões

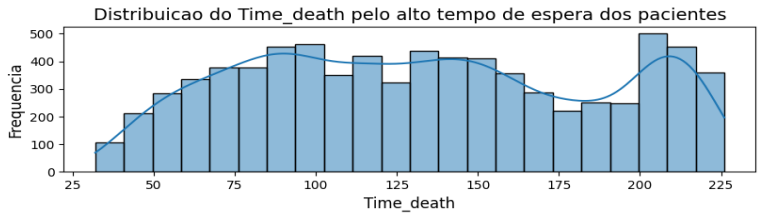
Visualização da importância das características identificadas pelo modelo e análise da distribuição dos pacientes com longos tempos de espera.

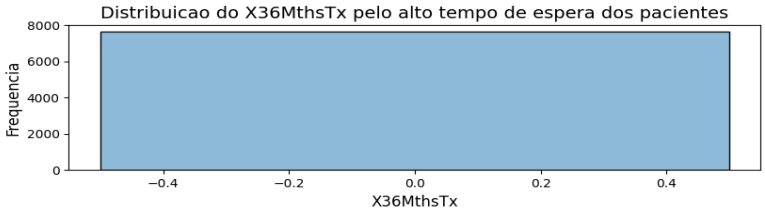
Código utilizado:

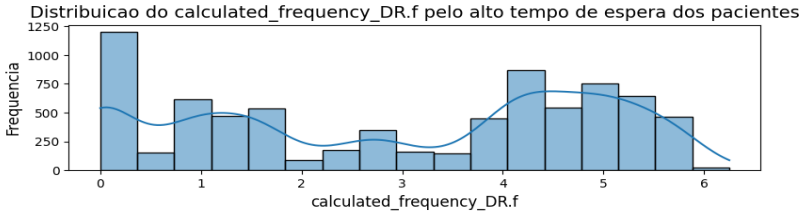
```
import matplotlib.pyplot as plt
import seaborn as sns

# pacientes com tempo de espera acima da mediana
alto_tempo_de_espera_indices = y_selected[y_selected == 1].index

# tempo de espera acima da mediana
alto_tempo_de_espera_data = df.loc[alto_tempo_de_espera_indices, selected_features]

# distribuicao para cada variavel
fig, axes = plt.subplots(nrows=len(selected_features), ncols=1, figsize=(10, 20))
fig.tight_layout(pad=8.0)

for i, feature in enumerate(selected_features):
    sns.histplot(alto_tempo_de_espera_data[feature], kde=True, ax=axes[i])
    axes[i].set_title(f'Distribuicao do {feature} pelo alto tempo de espera dos pacientes', fontsize=14)
    axes[i].set_xlabel(feature, fontsize=12)
    axes[i].set_ylabel('Frequencia', fontsize=12)

plt.show()
```

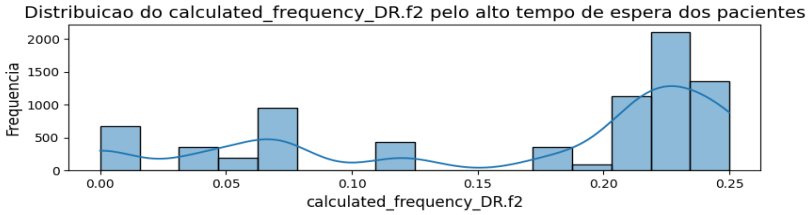
Output das visualizações:

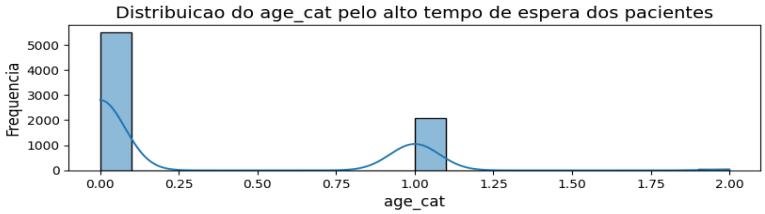












3.1.1. Discussão de resultados:

time_tx: a maioria desses pacientes possui valores altos nesta variável, indicando um padrão de tratamento prolongado antes do transplante.

time_death: a distribuição mostra uma gama variada, com alguns picos, indicando períodos específicos em que os pacientes que esperam mais tempo tendem a falecer.

x36mthstx: há uma concentração de valores mais baixos, sugerindo que menos pacientes atingem um tratamento prolongado de 36 meses.

calculated_frequency_dr.f e calculated_frequency_dr.f2: estas variáveis relacionadas ao cálculo de frequências especificas mostram distribuições variadas, indicando diferentes padrões de características genéticas.

3.2. Impacto Potencial

Eficiência Operacional: Melhoria na alocação de recursos e na priorização de pacientes, reduzindo o tempo médio de espera.

Políticas de Saúde: Informação baseada em dados que pode influenciar políticas públicas e práticas recomendadas para a gestão de listas de espera.

4. GITHUB

Durante o desenvolvimento deste projeto, utilizamos o GitHub como nossa principal plataforma de armazenamento e gerenciamento de versões. O repositório do projeto pode ser acessado através do seguinte link: https://github.com/vemado/PROJETO_APLICADO_II_51667_2024.1

4.1 Estrutura do Repositório:

- Código: O notebook Jupyter usado para análise de dados e modelagem

- Documentação: Documentos explicativos, relatórios parciais e a versão final do relatório técnico estão disponíveis para consulta.
- Dados: Conjuntos de dados utilizados no projeto foram armazenados com instruções detalhadas sobre como carregá-los e manipulá-los.