Projeto final de *DataMining*

Eduardo Marques
Eleonora Weiner
Vinícius Guerra
Vinicius Mattoso

Contextualização e Objetivo do Trabalho

Devido às transformações tecnológicas que vêm ocorrendo em nossa sociedade, o acesso aos dados oriundos do Sistema Único de Saúde (SUS) têm se tornado cada vez mais fácil. Nosso grupo, visando melhor entender quais são os principais fatores que levam um paciente a óbito, decidiu construir um modelo de classificação binária com as informações disponibilizadas pelo banco de dados abertos do SUS.

Compreender quais são os principais fatores associados ao óbito de pacientes internados, podem ser de grande utilidade para o planejamento de medidas preventivas a serem adotadas pelo próprio SUS. As seguradoras também possuem particular interesse sobre o tema, tendo em vista que a identificação de tais fatores e de sua representatividade para o óbito permitirá uma melhor avaliação de riscos, o que viabiliza a otimização dos processos de aquisição de novos clientes e de avaliação de solicitações de sinistro.

Etapas do Projeto

01
Aquisição dos dados

02
Preparação dos dados

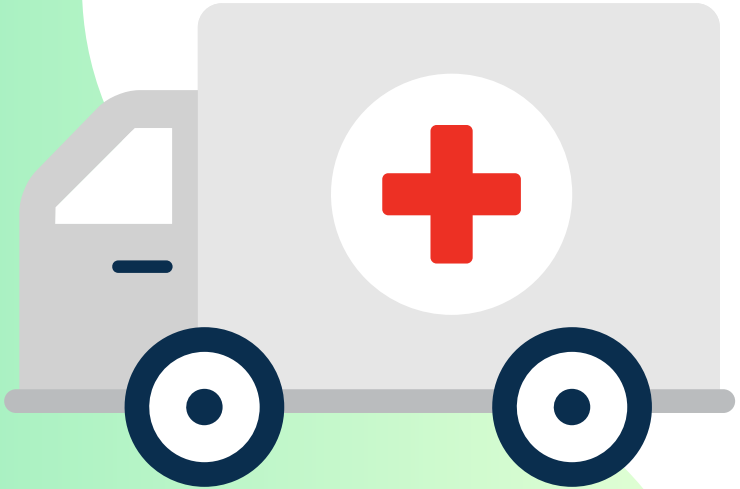
03
Criação do Baseline

04
Balanceamento e tratamento da base

05
Novos Modelos

06
Grid Search

07
Conclusões



01

Aquisição dos Dados

Para a aquisição dos dados foi utilizada a biblioteca PySUS

Pysus data

Nosso grupo optou por utilizar os dados do SUS do Rio de Janeiro reportados no primeiro mês do ano de 2019. Esse recorte foi feito para restringir o volume de dados e também evitar efeitos de desbalanceamento oriundos do período da pandemia COVID-19.

```
from pysus.online_data.SIH import download
df2 = download('RJ', 2019, month = 1)
df2.head()
```

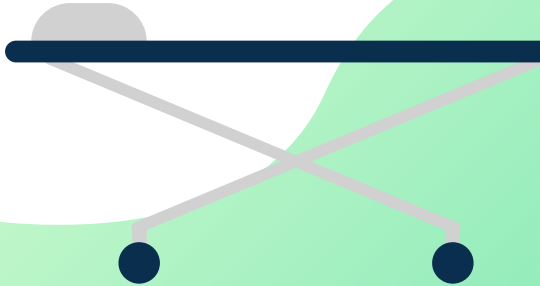
Para facilitar análises futuras, o arquivo baixado foi salvo em formato .csv

O arquivo contém 62412 linhas com 114 colunas.

02

Preparação dos Dados

Análise Exploratória | Filtro de "*Features*" | Data "*Encoding*"



Colunas da Base de Dados

A base de dados contém 114 colunas. Uma explicação mais detalhada de cada coluna pode ser encontrada no arquivo “IT_SIHSUS_1603.pdf” salvo no *github* do projeto.

Colunas que foram deletadas

- Dados do tipo *object* com informações de nomes, dados pessoais e informações de gestores
- Colunas vazias
- Informações gerenciais associadas ao hospital em que o paciente foi internado.
- Informações de datas que julgamos serem irrelevantes.

Tratamento de dados categóricos

As variáveis do tipo "*object*" que contêm os CID (Código Internacional de Doenças) dos diagnósticos dos pacientes foram codificadas para números para serem utilizadas em nossas análises.

Para evitar a criação de muitas colunas novas, foi utilizada a técnica do "*label encoding*", que faz a transformação de "*object*" para número sem criar novas colunas.

Instanciando o *encoder* para cada coluna.

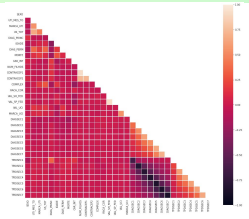
```
LABEL_ENCODING_DIAG1 = preprocessing.LabelEncoder()
LABEL_ENCODING_DIAG2 = preprocessing.LabelEncoder()
LABEL_ENCODING_DIAG3 = preprocessing.LabelEncoder()
LABEL_ENCODING_DIAG4 = preprocessing.LabelEncoder()
LABEL_ENCODING_DIAG5 = preprocessing.LabelEncoder()
LABEL_ENCODING_DIAG6 = preprocessing.LabelEncoder()
LABEL_ENCODING_DIAG7 = preprocessing.LabelEncoder()
```

Aplicando a transformação em cada coluna.

```
df2['DIAGSEC1']=LABEL_ENCODING_DIAG1.fit_transform(df2['DIAGSEC1'])
df2['DIAGSEC2']=LABEL_ENCODING_DIAG2.fit_transform(df2['DIAGSEC2'])
df2['DIAGSEC3']=LABEL_ENCODING_DIAG3.fit_transform(df2['DIAGSEC3'])
df2['DIAGSEC4']=LABEL_ENCODING_DIAG4.fit_transform(df2['DIAGSEC4'])
df2['DIAGSEC5']=LABEL_ENCODING_DIAG5.fit_transform(df2['DIAGSEC5'])
df2['DIAGSEC6']=LABEL_ENCODING_DIAG6.fit_transform(df2['DIAGSEC6'])
df2['DIAGSEC7']=LABEL_ENCODING_DIAG7.fit_transform(df2['DIAGSEC7'])
```

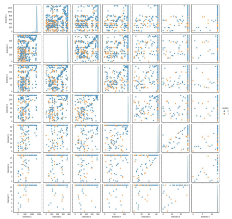
Análise Exploratória

Correlação entre colunas



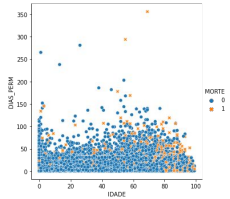
Avaliar as colunas que possuem grandes correlações com a coluna alvo (rótulo).

PairPlot de parte dos dados

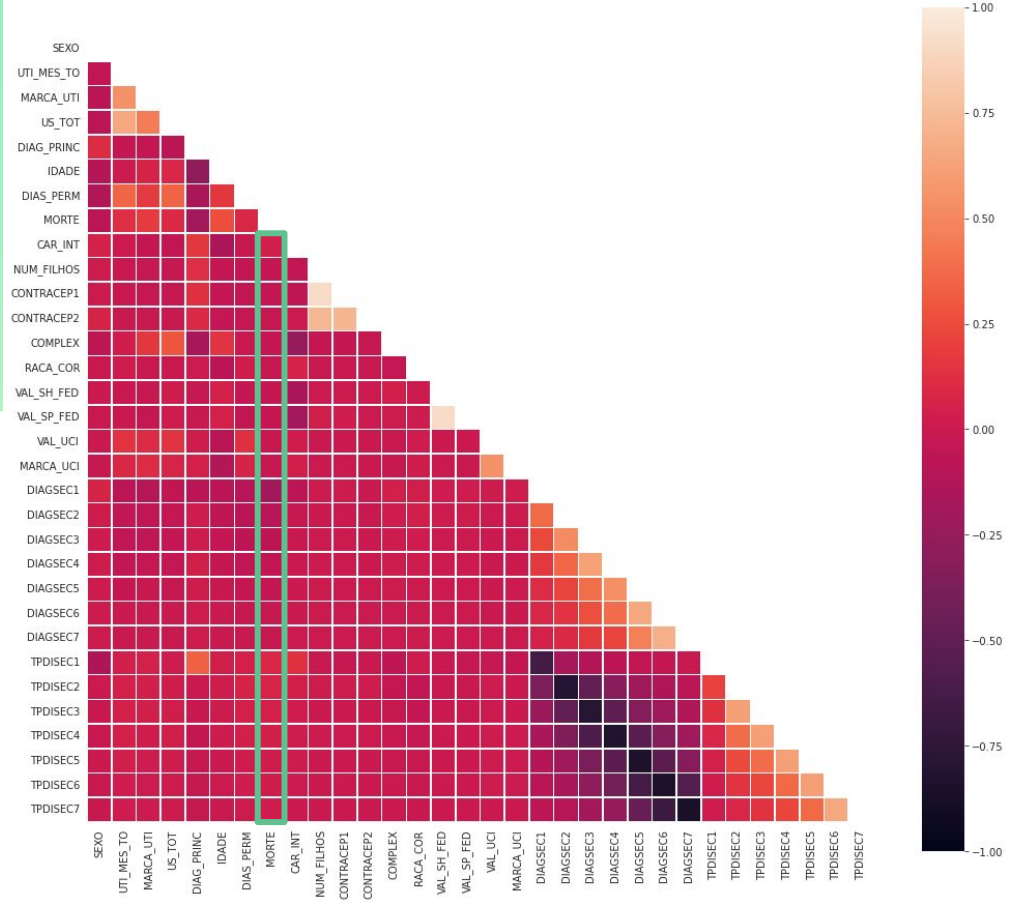


Avaliar graficamente se há correlações lineares entre as colunas.

Gráficos de dispersão segmentados

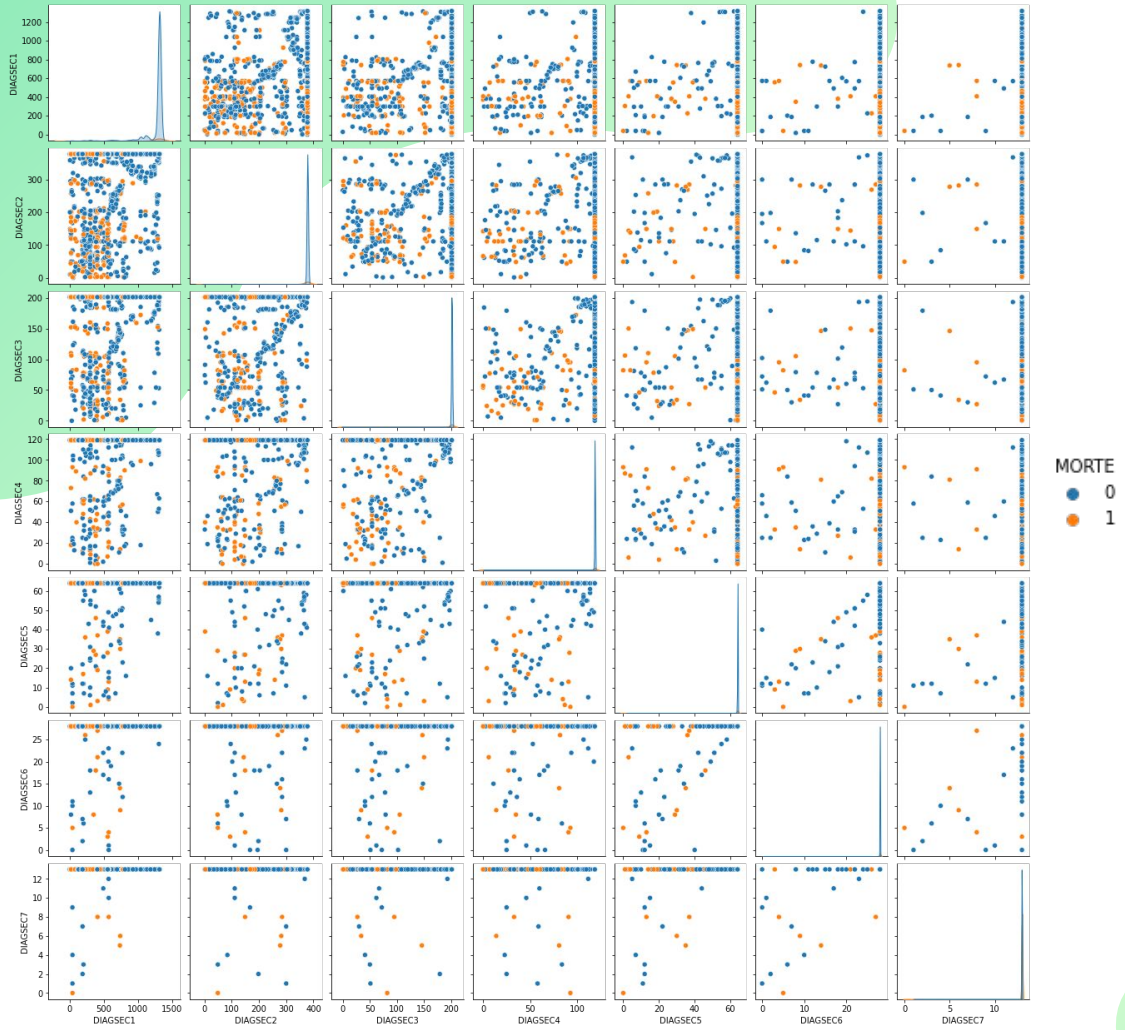


Avaliar se existe predominância de óbitos em algumas combinações de variáveis.



Correlação entre colunas

Observando a correlação das variáveis com a variável *target* (MORTE), podemos ver que não há nenhuma alta correlação específica. Para melhor compreender os dados, se faz necessária uma melhor análise gráfica.

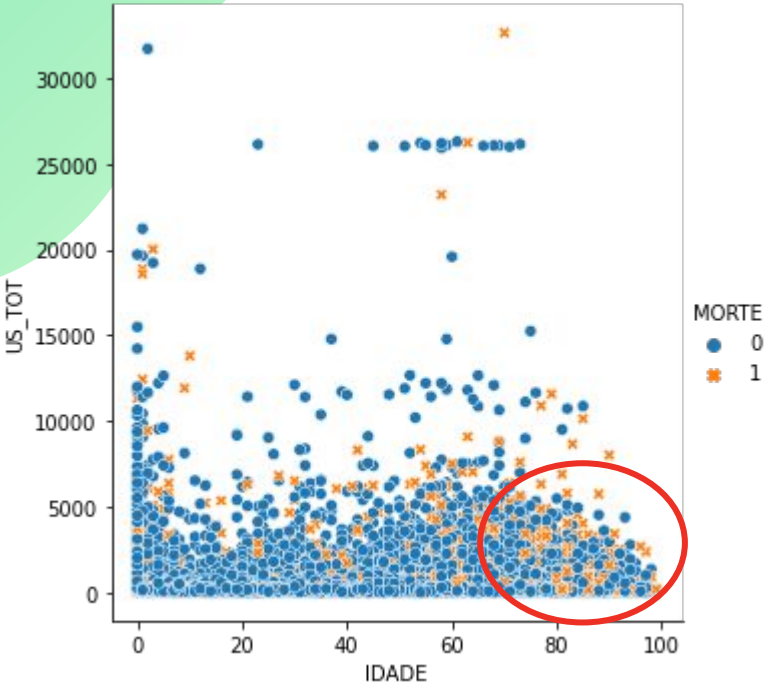


PairPlot de parte dos dados

Um gráfico muito interessante a ser analisado é o *pairplot*. Nesse gráfico é feita a análise da dispersão cruzada entre cada variável de interesse. No exemplo ao lado, fizemos a avaliação cruzada entre os diferentes diagnósticos recebidos pelo paciente no intuito de avaliar se existe uma correlação entre diagnósticos. Além disso, o gráfico foi segmentado pela variável *target*.

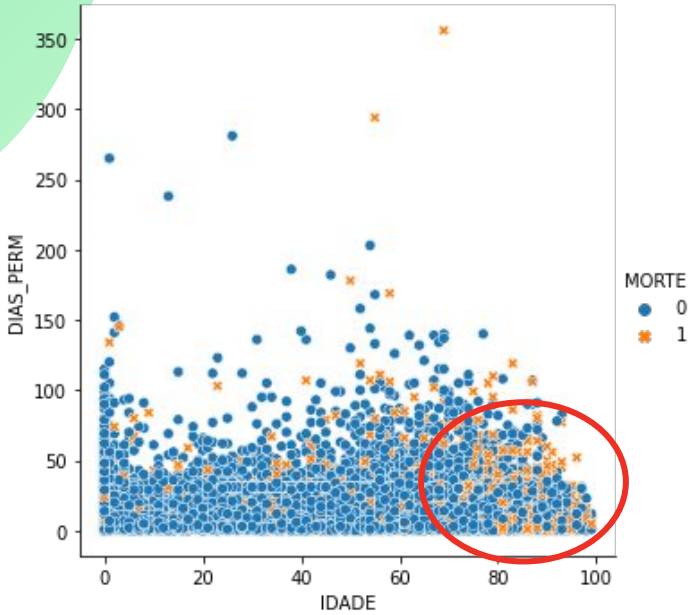
```
df_DIAG=df2[['DIAGSEC1','DIAGSEC2','DIAGSEC3',\
             'DIAGSEC4','DIAGSEC5','DIAGSEC6',\
             'DIAGSEC7','MORTE']].copy()
sns.pairplot(df_DIAG, hue="MORTE")
```

Gráficos de dispersão segmentados



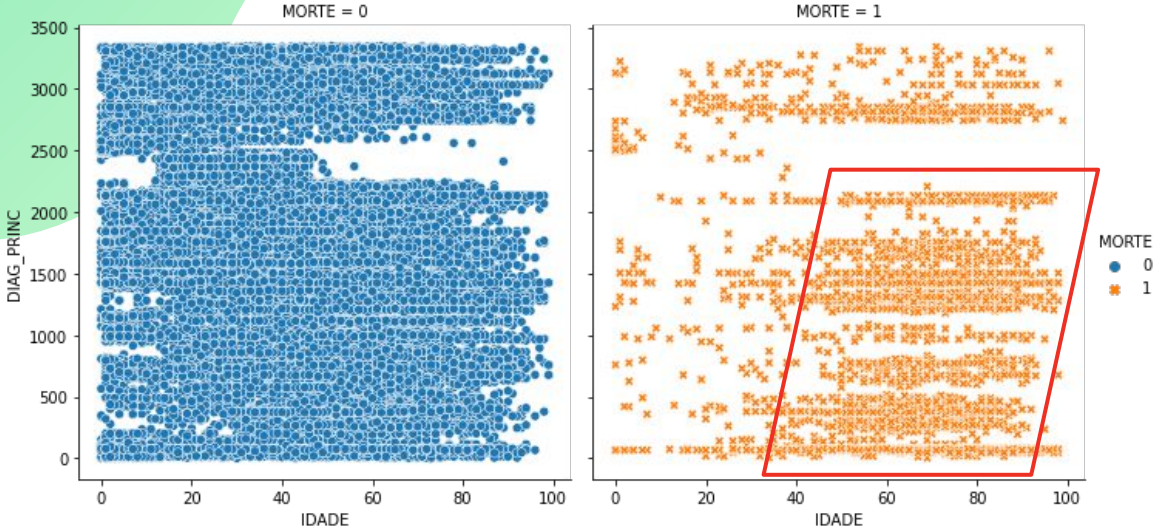
Analisando o custo total em dólar do paciente versus a sua idade, podemos observar que há uma concentração de óbitos em pacientes com idade maior que 70 anos e custo inferior a 5000 dólares.

Gráficos de dispersão segmentados



Essa mesma concentração é observada no gráfico dias de permanência versus idade, onde a idade maior que 70 anos revela um menor tempo de permanência e uma alta concentração de óbitos.

Gráficos de dispersão segmentados



O Diagnóstico Principal é preenchido com o CID (Código Internacional de Doenças). Para viabilizar a utilização desses dados foi feito o tratamento de *encoding*, para transformar as variáveis de *string* para número.

Podemos observar que pessoas com idades mais avançadas têm maior incidência de óbitos para vários CIDs diferentes.

03

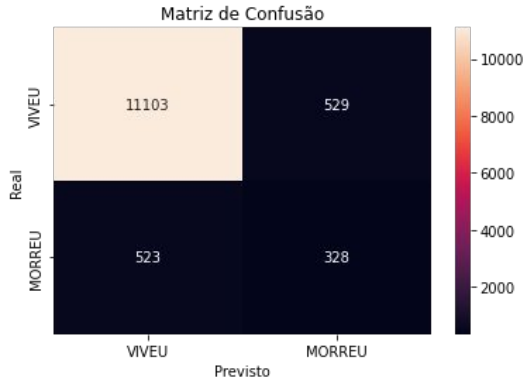
Criação do *Baseline*

Aplicação do modelo de árvore de decisão com parâmetros *default*

- Separação de dados de treino e teste;
- Treinamento do “*Decision Tree Classifier*”;
- Avaliação dos dados de teste.

0.9157

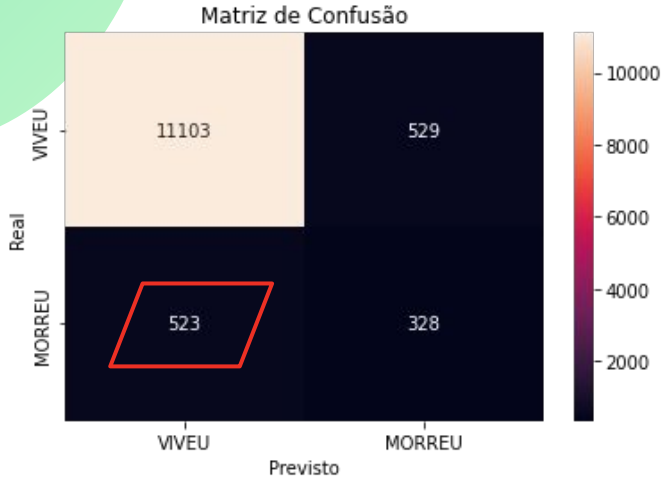
Acurácia dos dados de teste



0.3388 *Kappa*

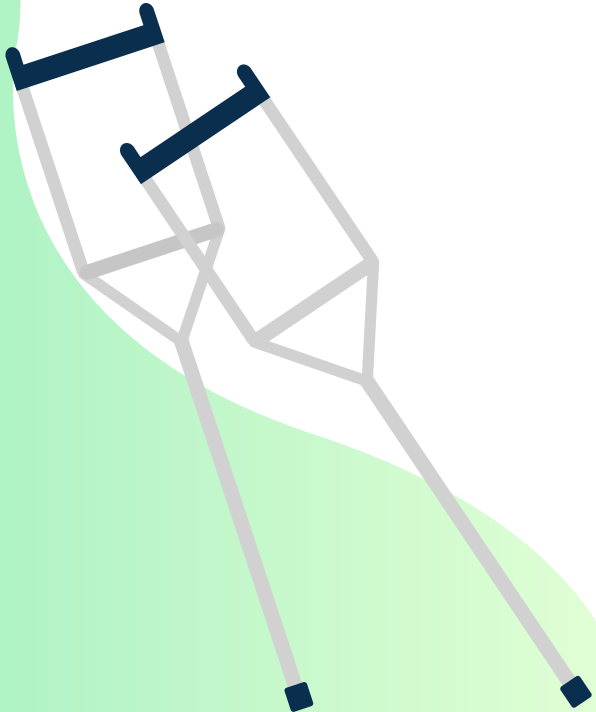
0.3841 *F1*

Modelo *Baseline*



Modelo *Baseline* se tornou muito bom no treino, mas com resultados não satisfatórios no teste, tendo em vista os parâmetros *Kappa* (0.338) e *F1-score* (0.3841).

Preocupação adicional de considerar um modelo onde existe um grande erro de previsão de sobrevivência (PREVISTO: “VIVEU”) para pacientes que acabam falecendo (REAL: “MORREU”).



04

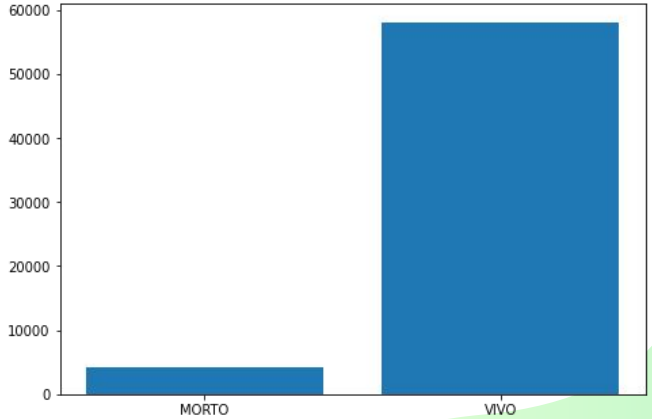
Balanceamento e tratamento da base de dados

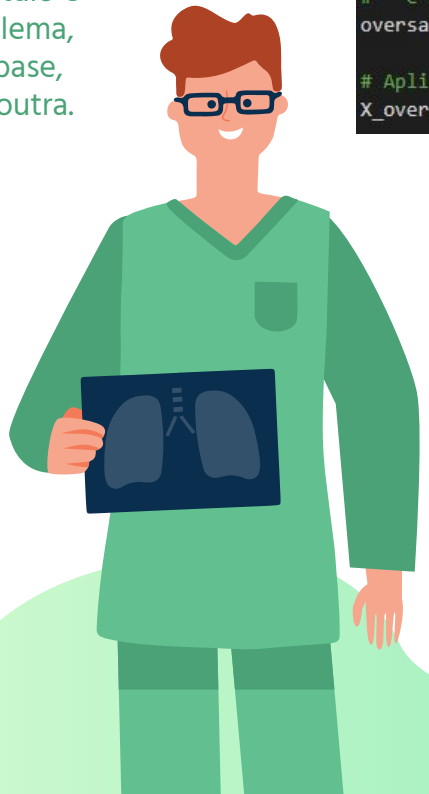
Balanceamento da base

O problema de desbalanceamento de base ocorre quando a quantidade de uma das classes do rótulo é muito superior a outra. Para resolver esse problema, aplicamos a estratégia de aumentar a menor base, mas sem deixar de ter uma predominância da outra.

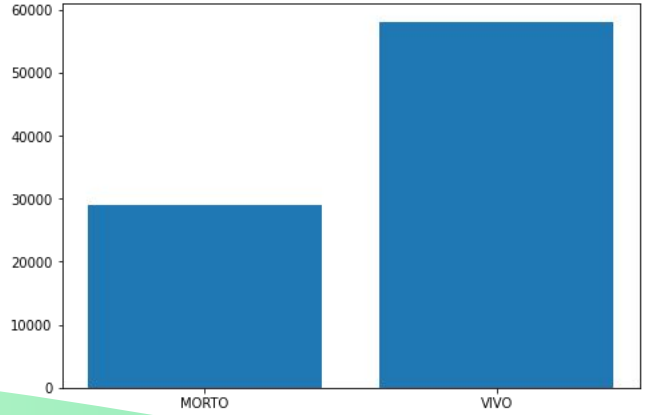
```python
from imblearn.over_sampling import RandomOverSampler
# Instanciar o random over sampler
#    e definir que vamos aumentar até ficar 1 morto para cada 2 vivos
oversample = RandomOverSampler(sampling_strategy=0.5)

# Aplicar o fit_resample em nossa base de dados
X_over, y_over = oversample.fit_resample(X, y)
```

Base Inicial





Nova base



Normalização da base

A normalização de dados é uma técnica estatística que visa reescalar os dados para que as colunas numéricas utilizadas pelo modelo tenham uma escala comum, sem distorcer as diferenças nos intervalos de valores nem perder informações.

Neste projeto, devido à aplicação da técnica do *encoder* para converter os CIDs em números, foi criada uma coluna com valores em uma escala extensa. Por exemplo, a idade vai até 99 anos enquanto o CID atinge a casa do milhar.
Sendo assim, a normalização é fundamental para a otimização do desempenho.

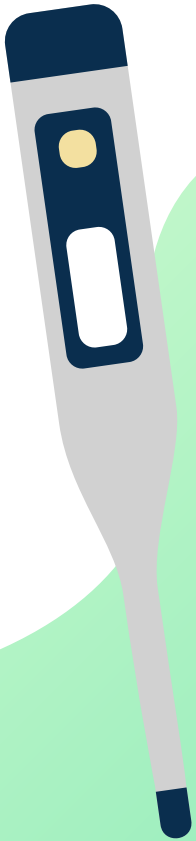
	SEXO	UTI_MES_TO	MARCA_UTI	US_TOT	DIAG_PRINC	IDADE	DIAS_PERM	CAR_INT
57240	1	0	0	25.31	3308	10	4	2
85818	1	0	0	86.24	1324	53	4	2

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler().fit(X_train_2)
X_train_2_norm = scaler.transform(X_train_2)
X_test_2_norm = scaler.transform(X_test_2)
```

05

Novos Modelos

Teste com outros modelos de classificação

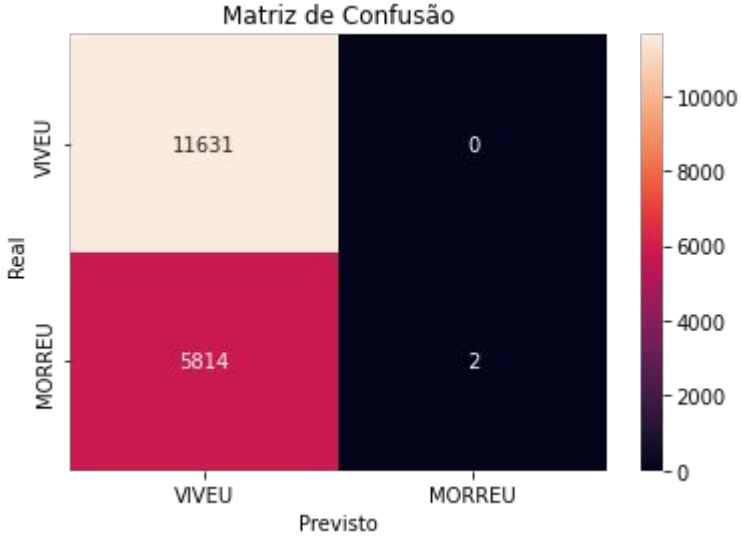


Decision Tree Classifier

Testamos novamente a árvore de decisão que foi utilizada para gerar o *baseline* após a aplicação das técnicas na base de dados.

0.667

Acurácia dos dados de teste com normalização e *resample*.

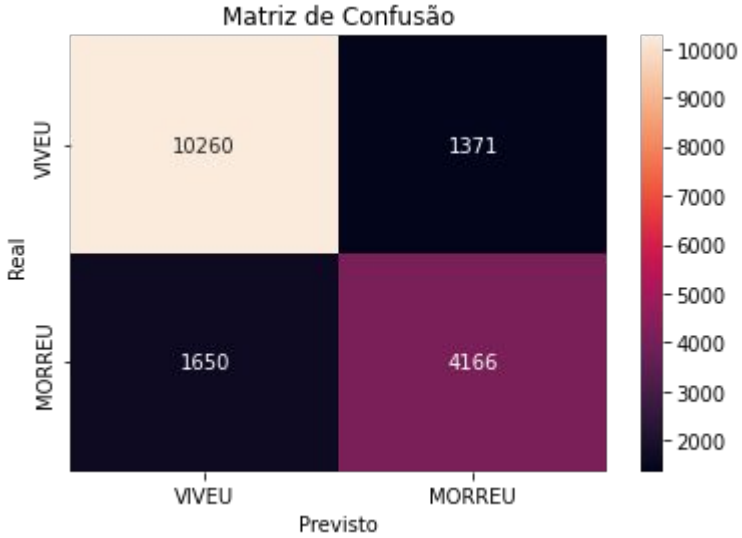


SVM (*Support Vector Machine*)

Esse método visa criar uma (reta/plano/hiperplano) que melhor separe o conjunto de variáveis.

0.827

Acurácia dos dados de teste com normalização e *resample*.

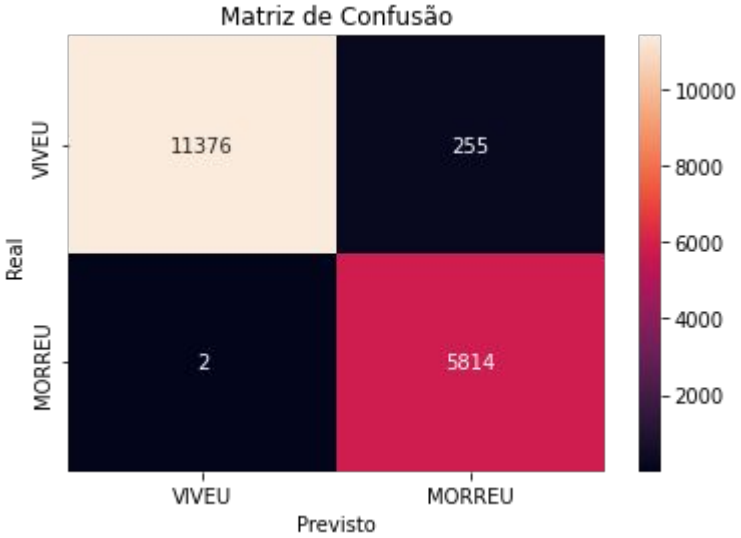


Random Forest Classifier

Método de aprendizado supervisionado que utiliza um comitê de árvores de decisão para melhor classificar individualmente cada paciente.

0.985

Acurácia dos dados de teste com normalização e *resample*.



06

GridSearch

Procurando melhorar o modelo de *Random Forest*

Best Model

```
from sklearn.model_selection import GridSearchCV

#dicionario com os parametros de tunning
tuned_parameters = [{'n_estimators': [50, 100, 150, 200],
                     'max_depth': [30, 50, 80, 100],
                     'criterion':['gini','entropy']}]

# Executar o grid search
model_RFC_GS = GridSearchCV(RandomForestClassifier( random_state=seed),\
                            tuned_parameters, scoring='f1')
model_RFC_GS.fit(X_train_2_norm, y_train_2);
```

Best Model

Acurácia do treino

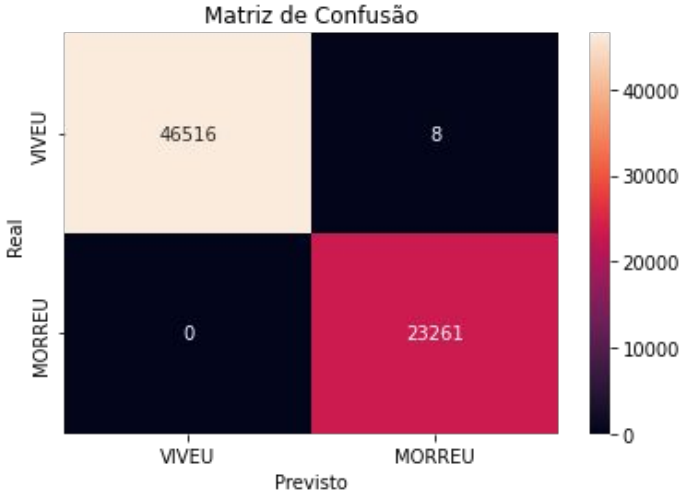
0.9998

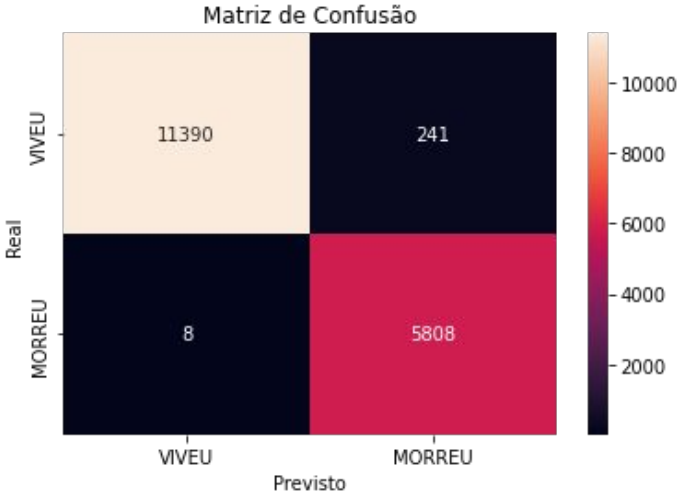
Kappa

0.9997

F1

0.9998





Acurácia do teste 0.986

Kappa 0.968

F1 0.979

Conclusões

Nesse projeto fomos capazes de criar um eficiente classificador binário. Durante o projeto reforçamos o conhecimento a respeito da necessidade de tratamento da base de dado para criação de um melhor modelo, e a importância de testar diferentes modelos e fazer o tunning dos parâmetros

Our Team

Eleonora Weiner

Eduardo Marques

Vinícius Guerra

Vinicius Mattoso

Thanks