Relatório competição 1

Lucas Vitor de Souza
Dezembro 2022

1 Análise exploratória dos dados

Os dados de treino foram utilizados para a verificação das métricas para medir o desempenho da classificação, contudo tanto a base de dados de treino como teste passaram pela mesma análise de dados antes de se iniciar o treinamento.

A princípio foram importados ambos os datasets, Treino e Teste, em formato .csv utilizando-se da biblioteca pandas. A primeira operação foi imprimir o dataset de treino para verificação das features, são 32 colunas com 227.122 linhas, tais quais variam seus tipos de dados em int64, float64 e object, como mostrado a seguir:

```
#   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   Unnamed: 0                  227122 non-null  int64
 1   NR_SEQ_REQUISICAO           227122 non-null  int64
 2   NR_SEQ_ITEM                 227122 non-null  int64
 3   DT_REQUISICAO               227122 non-null  int64
 4   DS_TIPO_GUIA                227122 non-null  object
 5   DT_NASCIMENTO               227112 non-null  float64
 6   NR_PRODUTO                  227122 non-null  int64
 7   DS_TIPO_PREST_SOLICITANTE   227122 non-null  object
 8   DS_CBO                      227122 non-null  object
 9   DS_TIPO_CONSULTA            10511 non-null   object
 10  QT_TEMPO_DOENCA             266 non-null     float64
 11  DS_UNIDADE_TEMPO_DOENCA     266 non-null     object
 12  DS_TIPO_DOENCA              531 non-null     object
 13  DS_INDICACAO_ACIDENTE       209539 non-null  object
 14  DS_TIPO_SAIDA               0 non-null       float64
 15  DS_TIPO_INTERNACAO          59863 non-null   object
 16  DS_REGIME_INTERNACAO        59863 non-null   object
 17  DS_CARATER_ATENDIMENTO      227122 non-null  object
 18  DS_TIPO_ACOMODACAO          59781 non-null   object
 19  QT_DIA_SOLICITADO           58995 non-null   float64
 20  CD_GUIA_REFERENCIA          37463 non-null   float64
 21  DS_TIPO_ATENDIMENTO         168045 non-null  object
 22  CD_CID                      131250 non-null  object
 23  DS_INDICACAO_CLINICA        179944 non-null  object
 24  DS_TIPO_ITEM                227122 non-null  object
 25  CD_ITEM                     227122 non-null  int64
 26  DS_ITEM                     227122 non-null  object
 27  DS_CLASSE                   227122 non-null  object
 28  DS_SUBGRUPO                 227122 non-null  object
 29  DS_GRUPO                    227122 non-null  object
 30  QT_SOLICITADA               227122 non-null  float64
 31  DS_STATUS_ITEM              227122 non-null  object
dtypes: float64(6), int64(6), object(20)
```

Os dados contidos nesse dataset podem ser divididos em Numéricos e Categóricos, com variáveis numericas tanto discretas quanto contínuas, pode ser observado também através dessa tabela, que existem várias features que possuem valores Null (NaN). Dessa forma foi necessária verificação da porcentagem de valores NaN em cada Feature:

```
Data columns (total 32 columns):
#   Column                      % NaN
    ------                      --------
Unnamed: 0                      0.000000
NR_SEQ_REQUISICAO               0.000000
NR_SEQ_ITEM                     0.000000
DT_REQUISICAO                   0.000000
DS_TIPO_GUIA                    0.000000
DT_NASCIMENTO                   0.004403
NR_PRODUTO                      0.000000
DS_TIPO_PREST_SOLICITANTE       0.000000
DS_CBO                          0.000000
DS_TIPO_CONSULTA               95.372091
QT_TEMPO_DOENCA                99.882882
DS_UNIDADE_TEMPO_DOENCA        99.882882
DS_TIPO_DOENCA                 99.766205
DS_INDICACAO_ACIDENTE           7.741654
DS_TIPO_SAIDA                 100.000000
DS_TIPO_INTERNACAO             73.642800
DS_REGIME_INTERNACAO           73.642800
DS_CARATER_ATENDIMENTO          0.000000
DS_TIPO_ACOMODACAO             73.678904
QT_DIA_SOLICITADO              74.024973
CD_GUIA_REFERENCIA             83.505341
DS_TIPO_ATENDIMENTO            26.011131
CD_CID                         42.211675
DS_INDICACAO_CLINICA           20.772096
DS_TIPO_ITEM                    0.000000
CD_ITEM                         0.000000
DS_ITEM                         0.000000
DS_CLASSE                       0.000000
DS_SUBGRUPO                     0.000000
DS_GRUPO                        0.000000
QT_SOLICITADA                   0.000000
DS_STATUS_ITEM                  0.000000
```

Para o tratamento desses dados do tipo NaN, foram removidas as Features nas quais as porcentagens desses valores ultrapassam 70%, para as outras Features, foram substituidos pelo tipo mais comum de cada, por exemplo na coluna `DS_TIPO_ATENDIMENTO` que contém 26,01% de dados NaN o tipo ‘Exames’ era o mais comum, com uma frequencia de 115826 ocorrências, como mostrado abaixo.

```
count     168045
unique        13
top       Exames
freq      115826
Name: DS_TIPO_ATENDIMENTO, dtype: object
```

O Passo seguinte foi fazer a remoção de algumas Features que não seriam utilizadas na análise , por se serem consideradas pouco relevantes na inferencia final da Classe (Autorizado, Não Autorizado), foram elas `"DS_ITEM"`, `"DS_STATUS_ITEM"`, `"DS_CLASSE"`, `"DT_NASCIMENTO"`. Com isso o nosso conjunto de dados ficou da seguinte maneira:

```
Data columns (total 16 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   NR_SEQ_REQUISICAO           227122 non-null  int64
 1   NR_SEQ_ITEM                 227122 non-null  int64
 2   DT_REQUISICAO               227122 non-null  int64
 3   DS_TIPO_GUIA                227122 non-null  object
 4   NR_PRODUTO                  227122 non-null  int64
 5   DS_TIPO_PREST_SOLICITANTE   227122 non-null  object
 6   DS_CBO                      227122 non-null  int64
 7   DS_INDICACAO_ACIDENTE       227122 non-null  object
 8   DS_CARATER_ATENDIMENTO      227122 non-null  object
 9   DS_TIPO_ATENDIMENTO         227122 non-null  object
 10  CD_CID                      227122 non-null  int64
 11  DS_TIPO_ITEM                227122 non-null  object
 12  CD_ITEM                     227122 non-null  int64
 13  DS_SUBGRUPO                 227122 non-null  int64
 14  DS_GRUPO                    227122 non-null  object
 15  QT_SOLICITADA               227122 non-null  float64
dtypes: float64(1), int64(8), object(7)
```

2 Pré-processamentos realizados

Com a análise exploratória dos dados realizada, faz-se necessário o pré-processamento dos dados que, como apresentado anteriormente, seriam as substituições dos valores NaN, remoção das features nas quais esses valores ultrapassam 70% da ocorência total dentre as variáveis e transformação das variáveis categóricas em discretas. Para realizar esse processamento foram utilizadas as seguintes funções e métodos:

```
# * Por ser uma feature importante DS_TIPO_ATENDIMENTO, os valores NaN foram substituidos por atendimentos do tipo EXAMES, que são os mais comuns.
# * Da mesma forma DS_INDICACAO_ACIDENTE, foi preenchido com Não Acidente os valores NaN, por serem mais comuns os tipos de atendimento "Não Acidente".
# * CD_CID foi atualizado os valores NaN para Z00(exame geral), por ser uma coluna importante para análise (diz qual a doença do paciente).
```

```
def process (dt):
     dt.DS_TIPO_ATENDIMENTO   =   dt.DS_TIPO_ATENDIMENTO.fillna('Exames',
inplace=False)
      dt.DS_INDICACAO_ACIDENTE   =   dt.DS_INDICACAO_ACIDENTE.fillna('N?o
acidente', inplace=False)
  dt.NR_PRODUTO = dt.NR_PRODUTO.fillna(1.0, inplace=False)
  dt.CD_CID = dt.CD_CID.fillna('Z00', inplace=False)


# Transformando Variáveis Categóricas em Discretas.


def categ_disc(dt):
  le = LabelEncoder()
  dt['DS_SUBGRUPO'] = le.fit_transform(dt['DS_SUBGRUPO'])
  dt['DS_CBO'] = le.fit_transform(dt['DS_CBO'])
  dt['CD_CID'] = le.fit_transform(dt['CD_CID'])


for cat_var in dt_train.select_dtypes(include='O').columns:
    le = LabelEncoder()
    le.fit(dt_train[cat_var])
    dt_train[cat_var + '_num'] = le.transform(dt_train[cat_var])
    dt_train.drop(cat_var, axis=1, inplace=True)
    dt_test[cat_var + '_num'] = le.transform(dt_test[cat_var])
    dt_test.drop(cat_var, axis=1, inplace=True)


# Removendo colunas com Valores Nulos


dt_train = dt_train.dropna(axis=1)
dt_test = dt_test.dropna(axis=1)


# Removendo colunas que não serão utilizadas.

dt_train = pd.DataFrame(dt_train.drop(["Unnamed: 0", "DS_ITEM",
"DS_STATUS_ITEM", "DS_CLASSE"], axis=1))
dt_test = pd.DataFrame(dt_test.drop(["Unnamed: 0", "DS_ITEM",
"DS_CLASSE", "DT_NASCIMENTO"], axis=1))
```

Com o pré-processamento aplicado, nosso dataset agora, fica com os seguintes tipos de dados:

```
Data columns (total 16 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   NR_SEQ_REQUISICAO               227122 non-null  int64
 1   NR_SEQ_ITEM                     227122 non-null  int64
 2   DT_REQUISICAO                   227122 non-null  int64
 3   NR_PRODUTO                      227122 non-null  int64
 4   DS_CBO                          227122 non-null  int64
 5   CD_CID                          227122 non-null  int64
 6   CD_ITEM                         227122 non-null  int64
 7   DS_SUBGRUPO                     227122 non-null  int64
 8   QT_SOLICITADA                   227122 non-null  float64
 9   DS_TIPO_GUIA_num                227122 non-null  int64
 10  DS_TIPO_PREST_SOLICITANTE_num   227122 non-null  int64
 11  DS_INDICACAO_ACIDENTE_num       227122 non-null  int64
 12  DS_CARATER_ATENDIMENTO_num      227122 non-null  int64
 13  DS_TIPO_ATENDIMENTO_num         227122 non-null  int64
 14  DS_TIPO_ITEM_num                227122 non-null  int64
 15  DS_GRUPO_num                    227122 non-null  int64
dtypes: float64(1), int64(15)
```

3 Configuração experimental

Para a realização tanto da análise dos dados, quanto da realização do pré-processamento dos dados, treinamento, teste, verificação de métricas e desempenho dos algoritmos testados, foi utilizada a linguagem de programação Python que é executada por dentro da ferramenta Colaboratory do Google na versão 3.8. As bibliotecas utilizadas, são de maioria providas pelo sklearn, contudo, também foi utilizada a biblioteca pandas, csv e matplotlib, para realização das análise, tratamentos e coleta de resultados dos nossos datasets. A seguir, todas as bibliotecas que foram utilizadas nas análises:

```
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.naive_bayes import GaussianNB, MultinomialNB, BernoulliNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score, ConfusionMatrixDisplay
from sklearn.preprocessing import StandardScaler, LabelEncoder
import pandas as pd
import csv
import matplotlib.pyplot as plt
```

4 Algoritmos utilizados

Os algoritmos utilizados no treinamento foram o Naive Bayes e o Randon Forest e comparados para verificar qual seria mais eficiente para a resolução do nosso problema.

O classificador Naive Bayes toma como premissa a suposição de independencia entre as variáveis do nosso dataset, esse modelo realiza uma classificação probabilística de observações.

P (A |B) = P (B | A) x P (A) / P (B)

Onde:

- P (B | A) significa a probabilidade de B acontecer já que o evento A se confirmou
- P (A) é a probabilidade de A acontecer
- P (B) é a probabilidade de B acontecer

Já com o classificador RandomForest serão criadas várias árvores de decisão, onde em cada uma delas acontecem os seguintes passos:

1 - seleção aleatória de algumas features,
2 - seleção da feature mais adequada para a posição de nó raiz,
3 - geração dos nós filhos.

Esses algoritmos de classificação podem ser instanciados da seguinte forma:

```
# Algoritimo RANDOM_FOREST.
clf = RandomForestClassifier(n_estimators=30, max_depth=None,
min_samples_split=2, random_state=3)

# Algoritimo NAIVE_BAYES.
clf = GaussianNB()

# Treinamento
clf.fit(x_train, y_train.values.ravel())
```

5 Resultados

Para análise das métricas e desempenho de cada classificador para o nosso conjunto de dados, foi utilizado o módulo metrics da biblioteca sklern, com ele conseguimos demonstrar para nossos algoritmos a Precisão, Recall, F1-Score, Acurácia e Matriz de confusão, tudo isso através das seguintes funções:

```
# Separação dos dados para TESTE de algoritmos.

x_train, x_test, y_train, y_test = train_test_split(dt_train,
dt_trainy, train_size=0.8, random_state=3)


# Verificação de Métricas para medir o desempenho da Classificação.

y_true, y_pred = y_test, clf.predict(x_test)
c = confusion_matrix(y_test, y_pred)
print(classification_report(y_true, y_pred))
print('\nAccuracy: {:.3f}%'.format(float(accuracy_score(y_test, y_pred)
* 100)))
print('\nConfusion Matrix:')
disp = ConfusionMatrixDisplay(confusion_matrix=c,
                              display_labels=clf.classes_)
disp.plot()
```

A precisão é a capacidade do classificador não rotular uma amostra negativa como positiva.

O recall é intuitivamente a habilidade do classificador em encontrar todas as amostras positivas.

A pontuação F-1 pode ser interpretada como uma média harmônica ponderada da precisão e da revocação, onde uma pontuação F-beta atinge seu melhor valor em 1 e pior pontuação em 0. Os pesos da pontuação F-beta lembram mais do que a precisão por um fator de beta. beta == 1.0 significa que o recall e a precisão são igualmente importantes.

Para a interpretação da Matriz de Confusão, levamos em consideração a seguinte análise:

Elemento na posição (0,0)TP: representa a quantidade de classificações Verdadeiros Positivos, ou seja, a quantidade de vezes que o modelo acertou a predição positiva conforme os dados reais. No exemplo, o classificador previu corretamente X casos em que o resultado foi AUTORIZADO.

Elemento na posição (0,1)FN: representa a quantidade de classificações Falsos Negativos, ou seja, a quantidade de vezes que o modelo previu incorretamente um resultado como

negativo. No exemplo, o classificador previu incorretamente Y casos em que o resultado foi NEGADO, sendo que a predição correta deveria ser AUTORIZADO.

Elemento na posição (1,0)FP: representa a quantidade de classificações Falsos Positivos, ou seja, a quantidade de vezes que o modelo previu incorretamente um resultado como positivo. No exemplo, o classificador previu incorretamente X casos em que o resultado foi AUTORIZADO, sendo que a predição correta deveria ser NEGADO.

Elemento na posição (1,1)TN: representa a quantidade de classificações Verdadeiros Negativos, ou seja, a quantidade de vezes que o modelo acertou a predição negativa conforme os dados reais. No exemplo, o classificador previu corretamente X casos em que o resultado foi NEGADO.

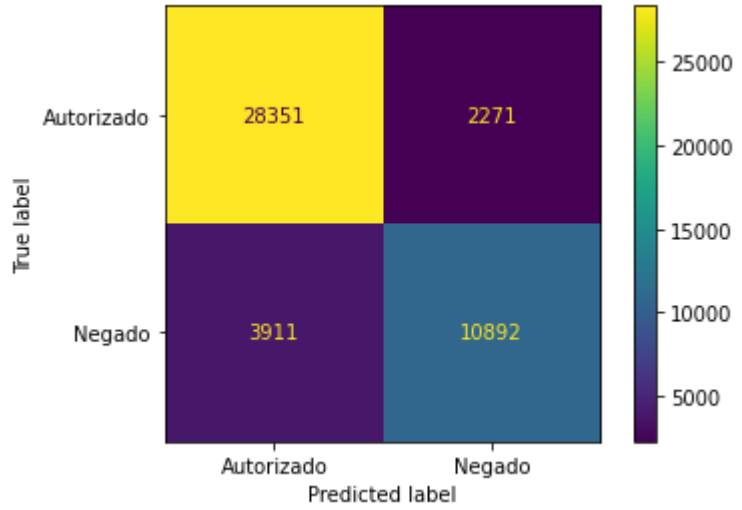
Logo abaixo, separado por algoritmos, temos os resultados obtidos após a aplicação de cada um dos classificadores apresentados anteriormente.

```
# Algoritmo RANDOM_FOREST.
```

	precision	recall	f1-score	support
Autorizado	0.88	0.93	0.90	30622
Negado	0.83	0.74	0.78	14803
accuracy			0.86	45425
macro avg	0.85	0.83	0.84	45425
weighted avg	0.86	0.86	0.86	45425

```
Accuracy: 86.391%
```

Confusion Matrix:

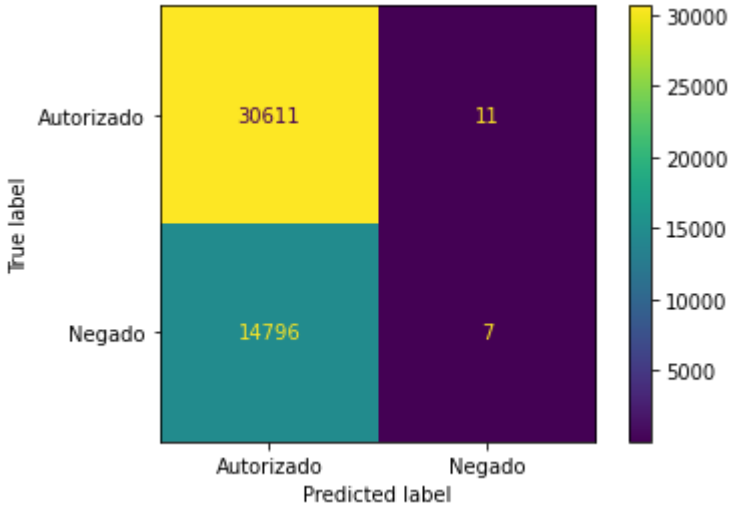


```
# Algoritmo NAIVE_BAYES.
```

	precision	recall	f1-score	support
Autorizado	0.67	1.00	0.81	30622
Negado	0.39	0.00	0.00	14803
accuracy			0.67	45425
macro avg	0.53	0.50	0.40	45425
weighted avg	0.58	0.67	0.54	45425

```
Accuracy: 67.403%
```

Confusion Matrix:



Dado os resultados a escolha como algoritmo mais eficiente para a classificação dos dados seria fácil, pois o RandomForest apresentou métricas bem mais eficientes do que o NaiveBayes, ao submeter o algoritimo do RandomForest no Kaggle, na página da competição, o score obtido foi de 0.70477. Contudo ao submeter o algoritmo do Naive Bayes, o score obtido na competição foi de 0.70701, alcançando uma pontuação maior do que a do RandomForest mesmo com métricas bastante confusas. Com isso a escolha mais segura para nosso dataset dentre os modelos de classificação é o algoritmo RandomForest.

6 Referências bibliográficas

[1] Vídeos teóricos e slides disponibilizados pela professora.
[2] https://towardsdatascience.com/understanding-random-forest-58381e0602d2
[3] https://www.ibm.com/cloud/learn/random-forest
[4] https://scikit-learn.org/stable/modules/naive_bayes.html
[5] https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c
[6]https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier
[7] https://dadosaocubo.com/analise-exploratoria-de-dados-com-python-parte-i/
[8]https://medium.com/data-hackers/entendendo-o-que-%C3%A9-matriz-de-confus%C3%A3o-com-python-114e683ec509