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 3

1. Bibliotecas usadas para o método analítico

Primeiro importamos as bibliotecas utilizadas para este Notebook.

```
from ucimlrepo import fetch_ucirepo
from IPython.display import display
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.multioutput import MultiOutputClassifier
from sklearn.linear_model import LinearRegression
from sklearn.multioutput import MultiOutputRegressor
from sklearn.metrics import accuracy_score
from matplotlib import pyplot as plt
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.ensemble import RandomForestClassifier
import pandas as pd
import seaborn as sns
```

2. Exibindo os datasets

Datasets e os dados neles presentes.

Exibindo os datasets puro

```
# fetch dataset
national_health_and_nutrition_health_survey_2013_2014_nhanes_age_prediction_subset = fetch_ucirepo(id=887)

# data (as pandas dataframes)
X = national_health_and_nutrition_health_survey_2013_2014_nhanes_age_prediction_subset.data.features
y = national_health_and_nutrition_health_survey_2013_2014_nhanes_age_prediction_subset.data.targets

# metadata
print(national_health_and_nutrition_health_survey_2013_2014_nhanes_age_prediction_subset.metadata)

# variable information
print(national_health_and_nutrition_health_survey_2013_2014_nhanes_age_prediction_subset.variables)
```

Python

```
{'uci_id': 887, 'name': 'National Health and Nutrition Health Survey 2013-2014 (NHANES) Age Prediction Subset', 'repository_url': 'https://archive.ics.uci.edu/dataset/887/national+health+and+nutrition+health+surv
         name     role        type demographic  \
0        SEQN       ID  Continuous        None
1   age_group   Target  Categorical         Age
2    RIDAGEYR    Other   Continuous         Age
3    RIAGENDR  Feature   Continuous      Gender
4      PAQ605  Feature   Continuous        None
5      BMXBMI  Feature   Continuous        None
6      LBXGLU  Feature   Continuous        None
7      DIQ010  Feature   Continuous        None
8      LBXGLT  Feature   Continuous        None
9       LBXIN  Feature   Continuous        None

                                    description units missing_values
0                       Respondent Sequence Number  None            no
1          Respondent's Age Group (senior/non-senior)  None            no
2                               Respondent's Age  None            no
3                            Respondent's Gender  None            no
4  If the respondent engages in moderate or vigor...  None            no
5                    Respondent's Body Mass Index  None            no
6            Respondent's Blood Glucose after fasting  None            no
7                      If the Respondent is diabetic  None            no
8                                 Respondent's Oral  None            no
9                 Respondent's Blood Insulin Levels  None            no
```

Linhas duplicadas:

```
[20]
# Verificacao de Dados ausentes
# Proporção de ausentes em cada atributo:
df_proporcao=df_health.isnull().sum()/len(df_health)

[21]
# Criacao de um DataFrame pandas com o resultado da operacao isnull()
df_isnull = df_health.isnull()
# Contabilizacao dos dados ausentes por atributo
print(df_health.isnull().sum())

...
sk                    0
faixaEtaria           0
idade                 0
sexo                  0
exerciciosSemanais    0
imc                   0
nivelGlicose          0
diabetico             0
LBXGLT                0
nivelInsulinaSangue   0
dtype: int64

[22]
# Verificacao de Linhas duplicadas
total_duplicados = df_health.duplicated(keep=False).sum()
print(f'Total de linhas duplicadas: {total_duplicados}')

...
Total de linhas duplicadas: 0
```

3. Gráficos obtidos

Gráfico de dispersão:

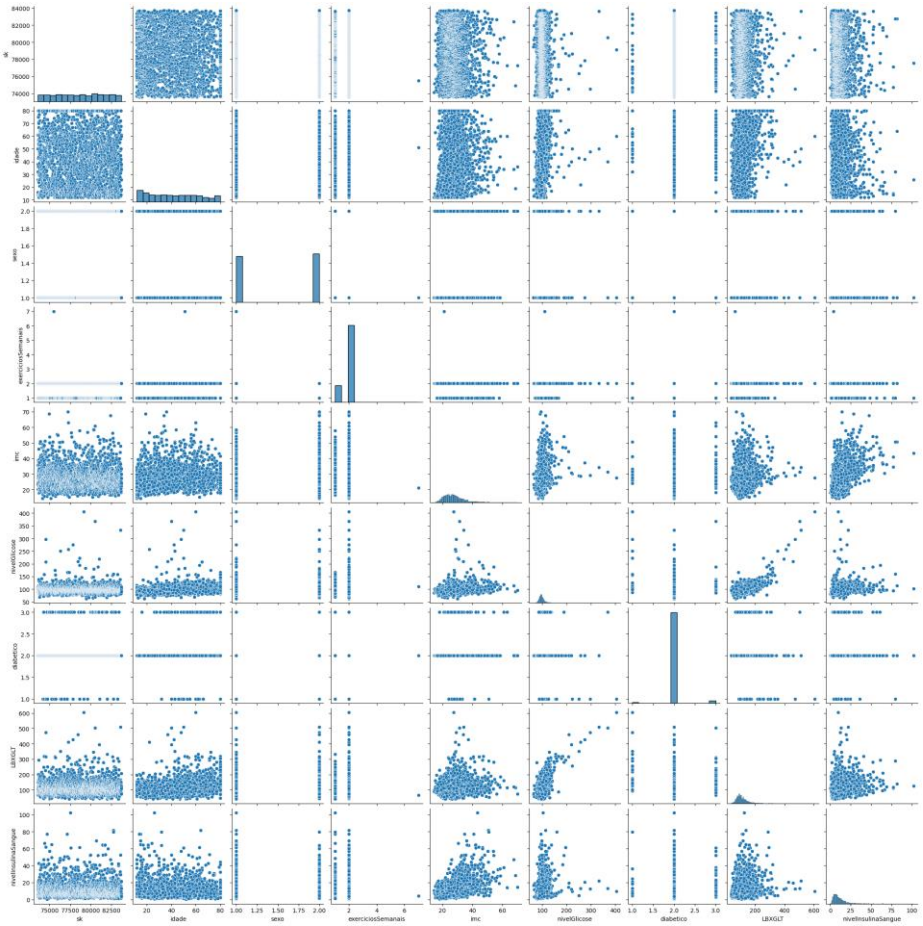
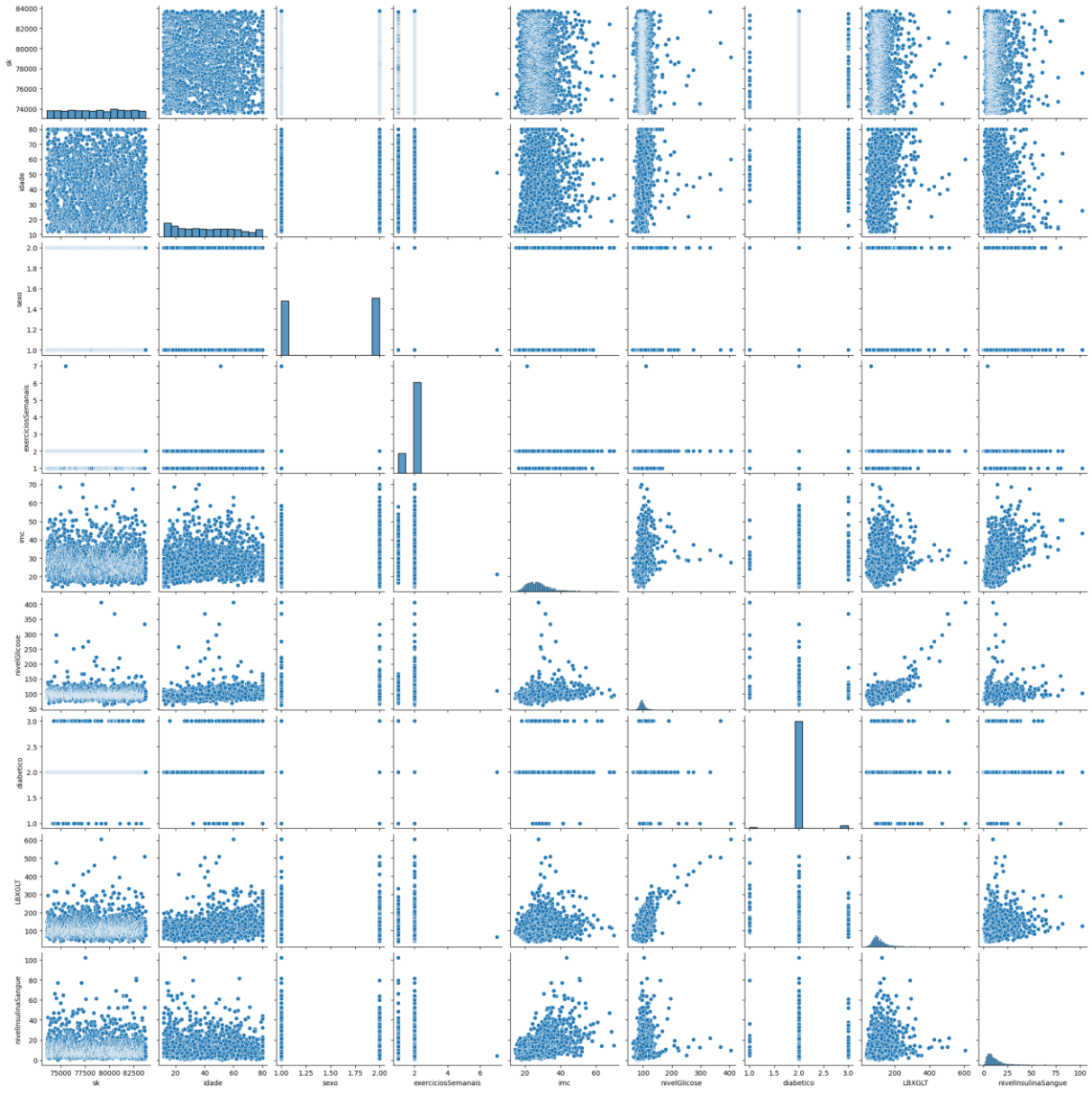


Gráfico Pair Plot:



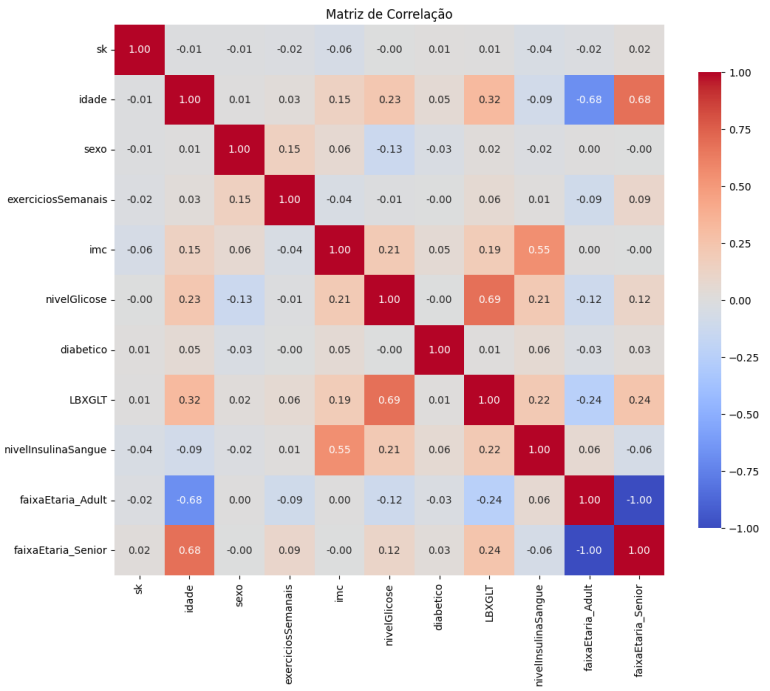
Matriz de correlação:

```python
# Correlação
# Usar one-hot encoding para colunas categóricas
df_encoded = pd.get_dummies(df_health, drop_first=False)
```

```python
# Calcular a matriz de correlação
correlation_matrix = df_encoded.corr()
#correlation_matrix = df_sample.corr()
print(correlation_matrix)
```

```
                          sk     idade      sexo  exerciciosSemanais  \
sk                 1.000000 -0.008806 -0.012962           -0.019701
idade             -0.008806  1.000000  0.006398            0.025973
sexo              -0.012962  0.006398  1.000000            0.151076
exerciciosSemanais -0.019701  0.025973  0.151076            1.000000
imc               -0.061343  0.147163  0.063873           -0.042935
nivelGlicose      -0.004147  0.229624 -0.132342           -0.007849
diabetico          0.014102  0.049970 -0.032769           -0.002599
LBXGLT             0.006036  0.318044  0.017406            0.060413
nivelInsulinaSangue -0.040028 -0.091879 -0.016660           0.010011
faixaEtaria_Adult  -0.018257 -0.684490  0.002767           -0.094789
faixaEtaria_Senior  0.018257  0.684490 -0.002767            0.094789

                           imc  nivelGlicose  diabetico    LBXGLT  \
sk                 -0.061343     -0.004147   0.014102  0.006036
idade               0.147163      0.229624   0.049970  0.318044
sexo                0.063873     -0.132342  -0.032769  0.017406
exerciciosSemanais -0.042935     -0.007849  -0.002599  0.060413
imc                 1.000000      0.208330   0.047133  0.193375
nivelGlicose        0.208330      1.000000  -0.004427  0.685579
diabetico           0.047133     -0.004427   1.000000  0.009796
LBXGLT              0.193375      0.685579   0.009796  1.000000
nivelInsulinaSangue  0.552717      0.211911   0.058986  0.217272
faixaEtaria_Adult    0.004147     -0.116462  -0.026399 -0.243113
faixaEtaria_Senior  -0.004147      0.116462   0.026399  0.243113
...
LBXGLT                          0.243113
nivelInsulinaSangue            -0.064159
faixaEtaria_Adult              -1.000000
faixaEtaria_Senior              1.000000
```

Mapa de calor da correlação:



Outliers:

```python
# Outliers
print(df_health.columns)
# Avaliacao de outiliers
# Para calcular múltiplos quantis (ex: 0.25, 0.5, 0.75)
print('Coluna com mais correlação com Senior x Adult = LBXGLT:')
quantis_coluna=df_health['LBXGLT'].quantile([0.25, 0.5, 0.75])
print(quantis_coluna)
resumo_coluna=df_health['LBXGLT'].describe()
print(resumo_coluna)
```

```
Index(['sk', 'faixaEtaria', 'idade', 'sexo', 'exerciciosSemanais', 'imc',
       'nivelGlicose', 'diabetico', 'LBXGLT', 'nivelInsulinaSangue'],
      dtype='object')
Coluna com mais correlação com Senior x Adult = LBXGLT:
0.25     87.0
0.50    105.0
0.75    130.0
Name: LBXGLT, dtype: float64
count    2278.000000
mean      114.978929
std        47.061239
min        40.000000
25%        87.000000
50%       105.000000
75%       130.000000
max       604.000000
Name: LBXGLT, dtype: float64
```

Regressão Logística:

```python
# define o scaler, prepara (aprende) e executa normalização
scaler = MinMaxScaler()
scaler.fit(X)
X=scaler.transform(X)
X_train, X_test, y_train, y_test=train_test_split(X, y, stratify=y, test_size=0.3, random_state=123)
```

30]

```python
# Logistic regression com grid search:
base_estimator = LogisticRegression(max_iter=1000, solver='liblinear', class_weight='balanced')
param_grid = {
    'C': [0.01, 0.1, 1, 10, 100],  # Regularização
    'penalty': ['l1', 'l2']         # Tipos de penalidade
}
```

31]

```python
# Configurando o GridSearchCV
clf = GridSearchCV(base_estimator, param_grid, cv=5, scoring='accuracy')
clf.fit(X_train, y_train)
```

32]

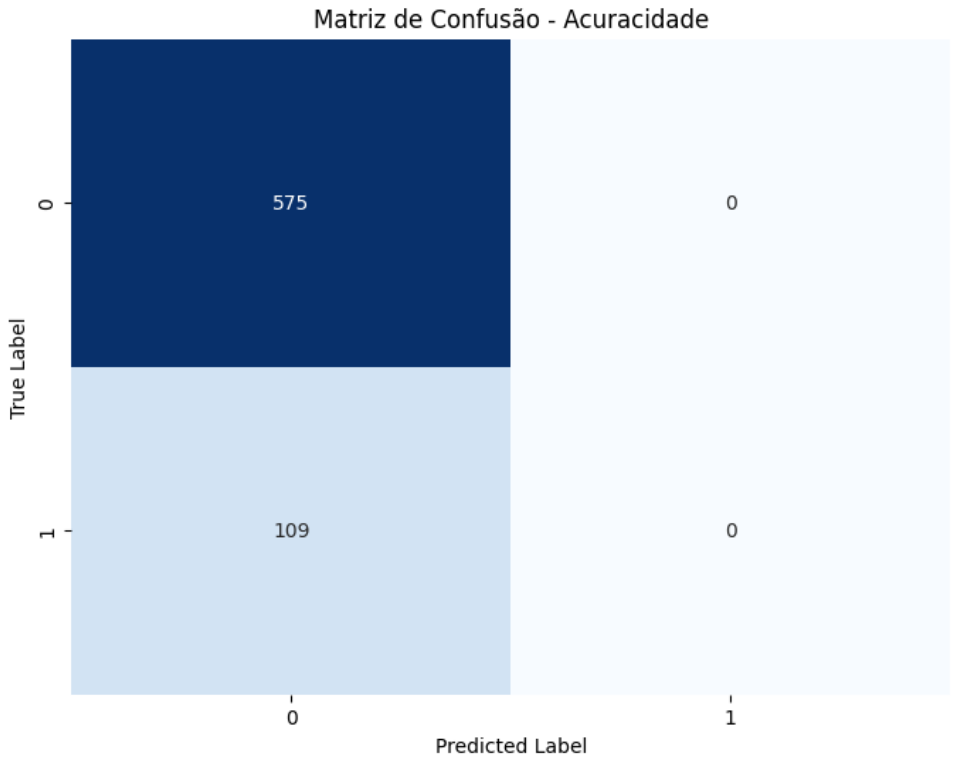
```
GridSearchCV
best_estimator_: LogisticRegression
LogisticRegression
```

4. Resultados e Acurácia

```
# Resultados
print("Melhores parâmetros:", clf.best_params_)
print("Acurácia no conjunto de teste:", clf.score(X_test, y_test))
print(clf.best_estimator_)
print('\nDetailed classification report:\n')
y_pred=clf.predict(X_test)
print(classification_report(y_test, y_pred, zero_division=0))
```

```
Melhores parâmetros: {'C': 0.01, 'penalty': 'l1'}
Acurácia no conjunto de teste: 0.8406432748538012
LogisticRegression(C=0.01, class_weight='balanced', max_iter=1000, penalty='l1',
                   solver='liblinear')

Detailed classification report:

              precision    recall  f1-score   support

       Adult       0.84      1.00      0.91       575
      Senior       0.00      0.00      0.00       109

    accuracy                           0.84       684
   macro avg       0.42      0.50      0.46       684
weighted avg       0.71      0.84      0.77       684
```

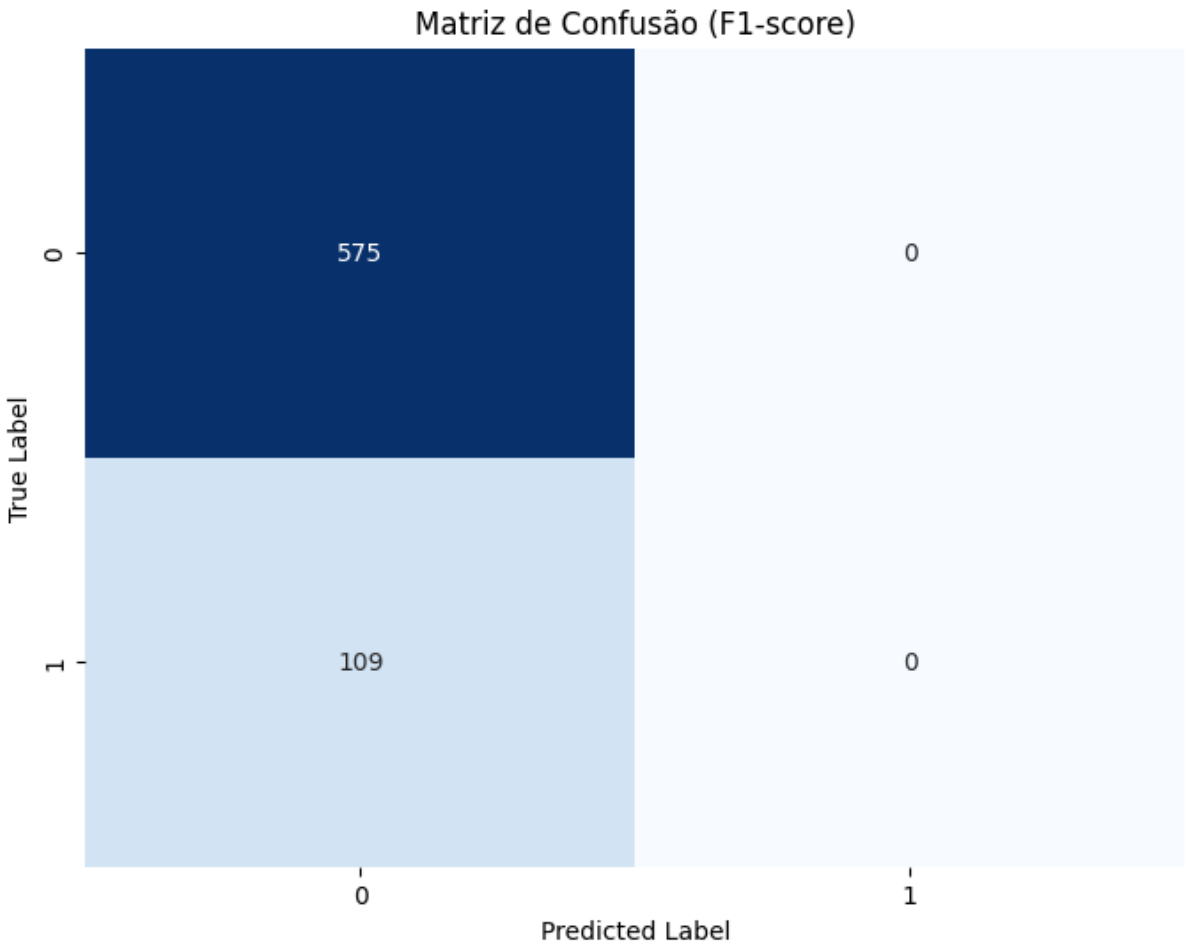
Matriz de confusão (acuracidade):



Conjunto de teste F1-score:

```python
# Exibindo os melhores parâmetros e o F1-score no conjunto de teste
print("Melhores parâmetros com base no F1-score:", clf_f1.best_params_)
print("F1-score no conjunto de teste:", clf_f1.score(X_test, y_test))
print(clf_f1.best_estimator_)

# Previsões
y_pred_f1 = clf_f1.predict(X_test)

# Cálculo da acurácia
acuracia = accuracy_score(y_test, y_pred_f1)
print("Acurácia no conjunto de teste:", acuracia)

# Relatório de classificação detalhado
print('\nDetailed classification report:\n')
print(classification_report(y_test, y_pred_f1, zero_division=0))
# Relatório de classificação detalhado
report = classification_report(y_test, y_pred_f1, zero_division=0, output_dict=True)

# Exibindo o relatório de classificação em formato de tabela
print('\nDetailed classification report:\n')
print(f"{'Classe':<10} {'Precisão':<10} {'Recall':<10} {'F1-score':<10} {'Suporte':<10}")
for classe, metrics in report.items():
    if classe != 'accuracy' and classe != 'macro avg' and classe != 'weighted avg':
        print(f"{classe:<10} {metrics['precision']:<10.2f} {metrics['recall']:<10.2f} {metrics['f1-score']:<10.2f} {metrics['support']:<10}")

# Exibindo a média
print(f"{'Média':<10} {report['macro avg']['precision']:<10.2f} {report['macro avg']['recall']:<10.2f} {report['macro avg']['f1-score']:<10.2f}")
```

```
[49]
...  Melhores parâmetros com base no F1-score: {'C': 0.01, 'penalty': 'l1'}
     F1-score no conjunto de teste: 0.7678631978410415
     LogisticRegression(C=0.01, class_weight='balanced', max_iter=1000, penalty='l1',
                        solver='liblinear')
     Acurácia no conjunto de teste: 0.8406432748538012

     Detailed classification report:

                   precision    recall  f1-score   support

            Adult       0.84      1.00      0.91       575
           Senior       0.00      0.00      0.00       109

         accuracy                           0.84       684
        macro avg       0.42      0.50      0.46       684
     weighted avg       0.71      0.84      0.77       684


     Detailed classification report:

     Classe     Precisão   Recall     F1-score   Suporte
     Adult      0.84       1.00       0.91       575.0
     Senior     0.00       0.00       0.00       109.0
     Média      0.42       0.50       0.46
```

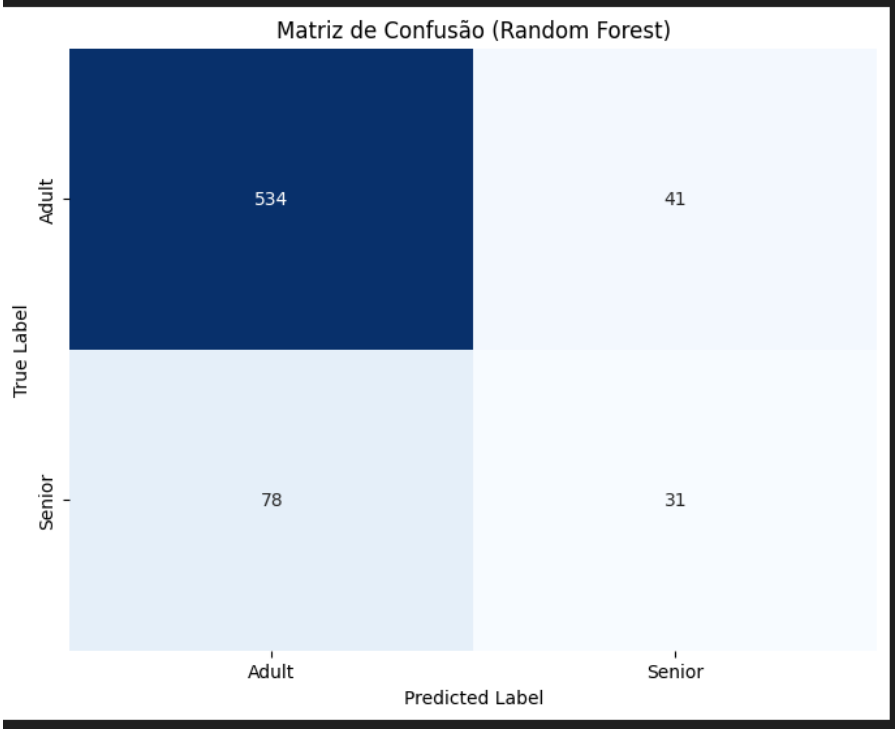


Random forest:

```
# random forest:

rf = RandomForestClassifier(class_weight='balanced')  # Ajustar o peso das classes
param_grid_rf = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
}

clf_rf = GridSearchCV(rf, param_grid_rf, cv=5, scoring='f1_weighted')
clf_rf.fit(X_train, y_train)

print("Melhores parâmetros com base no F1-score:", clf_rf.best_params_)
print("F1-score no conjunto de teste:", clf_rf.score(X_test, y_test))

y_pred_rf = clf_rf.predict(X_test)
print(classification_report(y_test, y_pred_rf, zero_division=0))
# Cálculo da acurácia
acuracia_rf = accuracy_score(y_test, y_pred_rf)
print("Acurácia no conjunto de teste:", acuracia_rf)

# Relatório de classificação detalhado
report_rf = classification_report(y_test, y_pred_rf, zero_division=0, output_dict=True)

# Exibindo o relatório de classificação em formato de tabela
print('\nDetailed classification report:\n')
print(f"{'Classe':<10} {'Precisão':<10} {'Recall':<10} {'F1-score':<10} {'Suporte':<10}")
for classe, metrics in report_rf.items():
    if classe not in ['accuracy', 'macro avg', 'weighted avg']:
        print(f"{classe:<10} {metrics['precision']:<10.2f} {metrics['recall']:<10.2f} {metrics['f1-score']:<10.2f} {metrics['support']:<10}")

# Exibindo a média
print(f"{'Média':<10} {report_rf['macro avg']['precision']:<10.2f} {report_rf['macro avg']['recall']:<10.2f} {report_rf['macro avg']['f1-score']:<10.2f} {'':<10}")

# Exibindo a acurácia como uma linha separada
print(f"{'Acurácia':<10} {acuracia_rf:<10.2f} {'':<10} {'':<10} {'':<10}")
```

```
Melhores parâmetros com base no F1-score: {'max_depth': 30, 'min_samples_split': 5, 'n_estimators': 200}
F1-score no conjunto de teste: 0.8079068365675973
              precision    recall  f1-score   support

       Adult       0.86      0.96      0.91       575
      Senior       0.47      0.20      0.28       109

    accuracy                           0.84       684
   macro avg       0.67      0.58      0.59       684
weighted avg       0.80      0.84      0.81       684

Acurácia no conjunto de teste: 0.8362573099415205

Detailed classification report:

Classe     Precisão   Recall     F1-score   Suporte
Adult      0.86       0.96       0.91       575.0
Senior     0.47       0.20       0.28       109.0
Média      0.67       0.58       0.59
Acurácia   0.84
```

```python
# matriz de confusao no random forest
y_pred_rf = clf_rf.predict(X_test)
conf_matrix_rf = confusion_matrix(y_test, y_pred_rf)
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix_rf, annot=True, fmt="d", cmap="Blues", cbar=False,
            xticklabels=['Adult', 'Senior'], yticklabels=['Adult', 'Senior'])
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Matriz de Confusão (Random Forest)")
plt.show()
# Relatório de classificação
print(classification_report(y_test, y_pred_rf, zero_division=0))
```



	precision	recall	f1-score	support
Adult	0.87	0.93	0.90	575
Senior	0.43	0.28	0.34	109
accuracy			0.83	684
macro avg	0.65	0.61	0.62	684
weighted avg	0.80	0.83	0.81	684

5. **Esboço do storytelling**

   Este projeto analisa um dataset de saúde, buscando prever a idade dos participantes com base em variáveis como hábitos alimentares e condições de saúde. Conseguimos analisar e correlacionar diversas variáveis de acordo com cada participante, tendo uma visão precisa da correlação com a idade. Além das tabelas, os gráficos nos permitirão ter uma visão mais concisa do que obtivemos com a análise.

   Os resultados revelam não apenas a precisão das previsões, mas também insights sobre como variáveis específicas se relacionam com a idade.

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

Link para acesso ao GitHub:

https://github.com/pcmassonjr/ProjetoAplicado2

7. Cronograma de Atividades