Documentação Técnica – Modelo de Avaliação de Risco Cardíaco

Este documento apresenta a documentação técnica completa do notebook Jupyter utilizado no desenvolvimento de um modelo de Machine Learning para avaliação do risco de doenças cardíacas. Todos os trechos de código estão incorporados ao texto e acompanhados de explicações técnicas.

1. Visão Geral do Projeto

O projeto segue um pipeline clássico de Machine Learning supervisionado: carregamento de dados, pré-processamento, separação em treino e teste, treinamento do modelo e avaliação de desempenho.

2. Justificativa do Algoritmo – Regressão Logística

A Regressão Logística foi escolhida por ser um algoritmo adequado à classificação binária, amplamente utilizado na área da saúde. O modelo fornece probabilidades de risco, possui alta interpretabilidade e baixo custo computacional, características essenciais em contextos médicos.

3. Pipeline Técnico com Código

Importação das bibliotecas

```
from sklearn.model_selection  import train_test_split, GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.metrics import classification_report, confusion_matrix,
roc_auc_score, RocCurveDisplay, roc_curve, accuracy_score, recall_score
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB, BernoulliNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
import pandas as pd
import numpy as np
import matplotlib.pylab as plt
import seaborn as sns
import shap
import random
```

Leitura dos dados

```
data_frame = pd.read_csv("cardio_train.csv", sep = ";")
```

Pré-processamento

Inicialmente, foi realizada a verificação de valores nulos na base de dados, a fim de garantir a consistência das informações utilizadas no treinamento do modelo.

```
data_frame.isnull().sum()
```

Em seguida, a coluna id foi removida por se tratar apenas de um identificador único, sem relevância preditiva, e procedeu-se à remoção de registros duplicados.

```
data_frame.drop("id", axis=1 ).duplicated().sum()
df_sem_duplicadas = data_frame.drop("id", axis=1).drop_duplicates()
```

Para assegurar a qualidade dos dados clínicos, foram aplicadas regras de validação sobre os valores de pressão arterial sistólica (ap_hi) e diastólica (ap_lo), mantendo apenas observações dentro de intervalos fisiologicamente plausíveis.

```
df_pradonizado = df_sem_duplicadas[
    (df_sem_duplicadas["ap_hi"].between(30, 280)) &
    (df_sem_duplicadas["ap_lo"].between(20, 200))
].copy()
```

Por fim, algumas variáveis foram ajustadas para tipos de dados adequados, garantindo compatibilidade com os algoritmos de aprendizado de máquina.

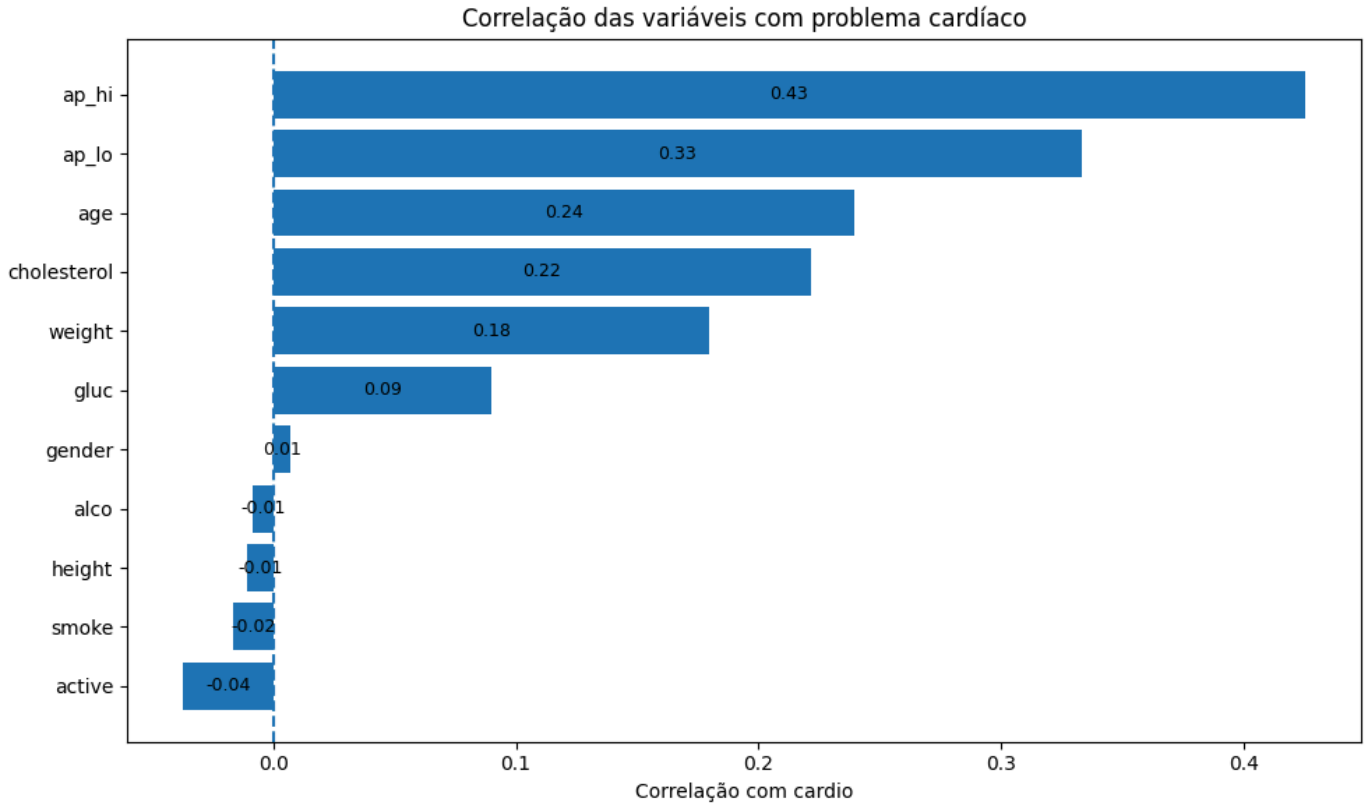
```
df_pradonizado["weight"] = df_pradonizado["weight"].astype(int)
df_pradonizado["smoke"] = df_pradonizado["smoke"].astype(bool)
df_pradonizado["alco"] = df_pradonizado["alco"].astype(bool)
df_pradonizado["active"] = df_pradonizado["active"].astype(bool)
df_pradonizado["cardio"] = df_pradonizado["cardio"].astype(bool)
```

Exploração dos dados

A análise exploratória teve como objetivo compreender a relação entre as variáveis explicativas e a presença de doença cardíaca. Inicialmente, foi avaliada a correlação entre cada variável e a variável alvo (cardio), permitindo identificar atributos com maior associação ao risco cardíaco.

```
features = df_pradonizado.drop(columns=["cardio"])

correlacoes = features.corrwith(df_pradonizado["cardio"])
correlacoes = correlacoes.sort_values()

plt.figure(figsize=(10, 6))
bars = plt.barh(correlacoes.index, correlacoes.values)

plt.xlabel("Correlação com cardio")
plt.title("Correlação das variáveis com problema cardíaco")

plt.axvline(0, linestyle="--")
```

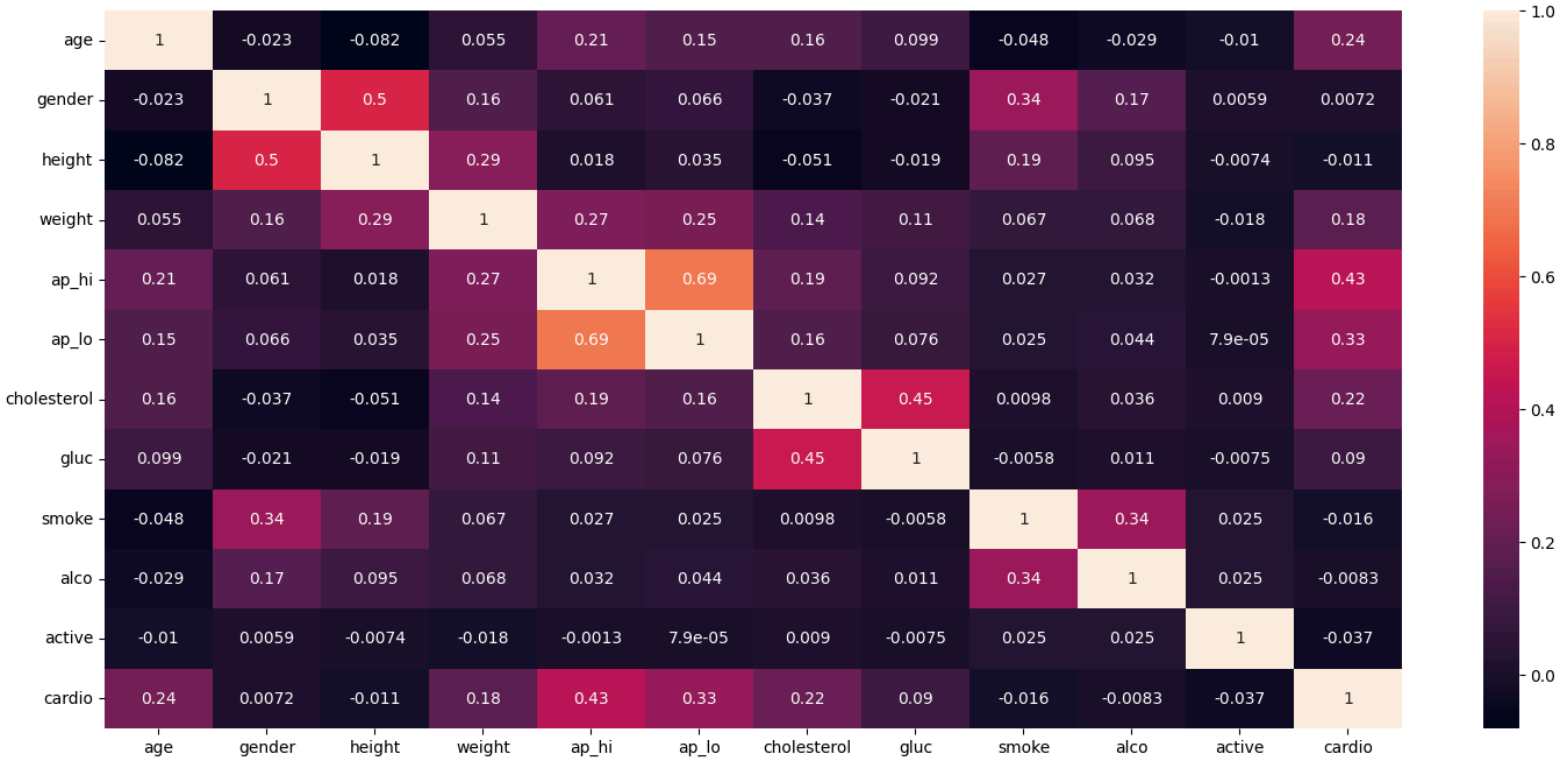
```
for bar in bars:
    valor = bar.get_width()
    y = bar.get_y() + bar.get_height() / 2

    plt.text(
        valor / 2,
        y,
        f"{valor:.2f}",
        va="center",
        ha="center",
        fontsize=9,
        color="black"
    )
```




```
plt.tight_layout()
plt.show()
```

Em complemento, foi utilizado um mapa de calor para visualizar correlações entre todas as variáveis, auxiliando na identificação de possíveis redundâncias e relações lineares relevantes.

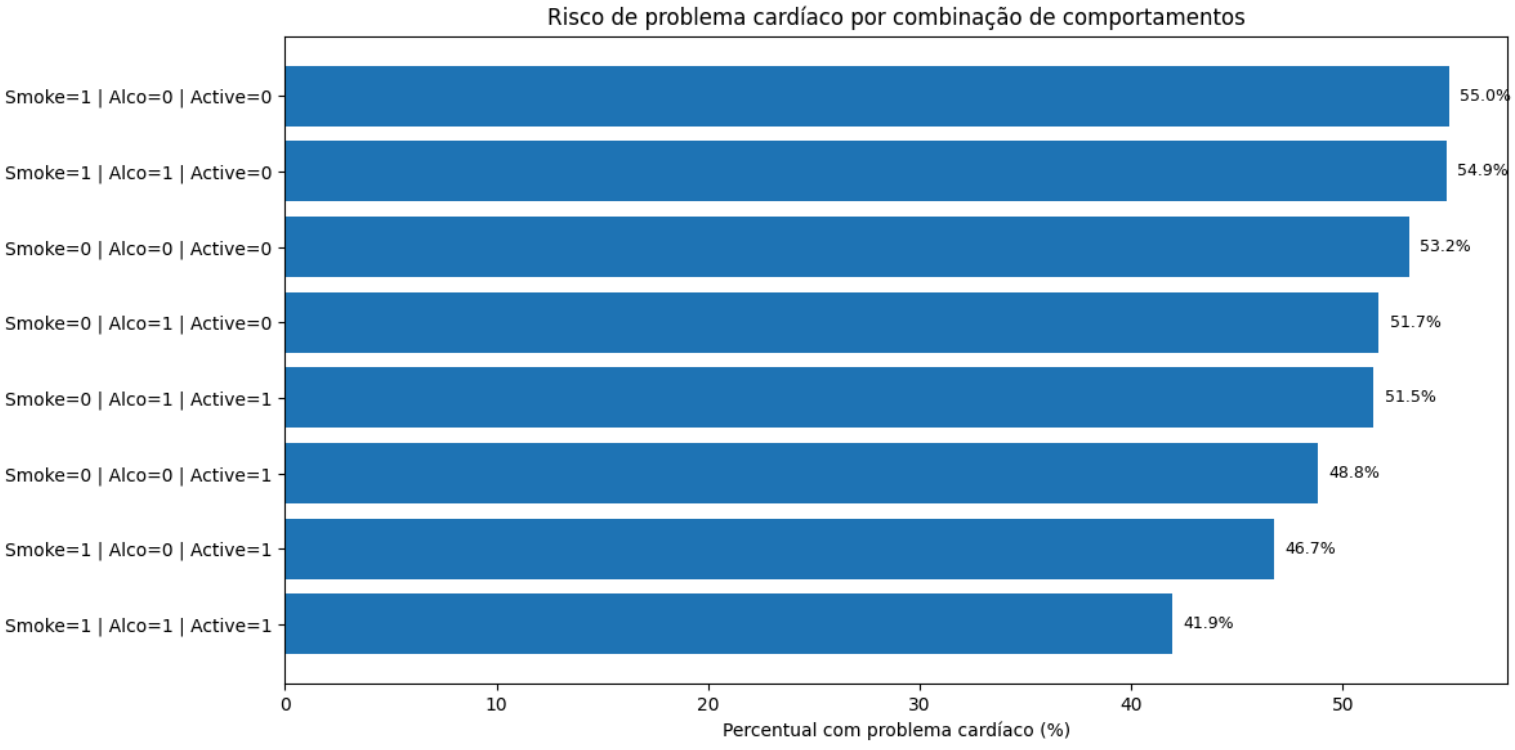
```
plt.rcParams["figure.figsize"] = (18,8)
ax = sns.heatmap(df_pradonizado.corr(), annot = True)
```



Adicionalmente, foram construídas visualizações específicas para investigar o impacto de comportamentos como tabagismo, consumo de álcool e atividade física, bem como a distribuição do risco cardíaco por faixas etárias, evidenciando padrões importantes para a modelagem.

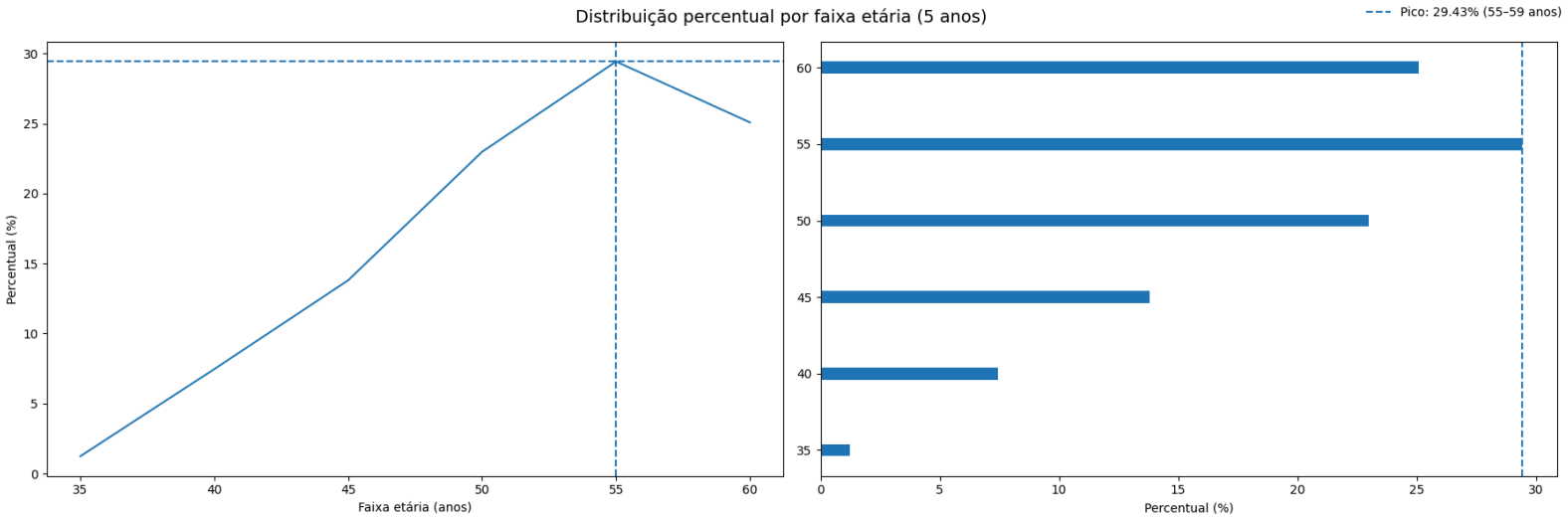
```
df = df_pradonizado.copy()

cols_binarias = ["smoke", "alco", "active", "cardio"]
df[cols_binarias] = df[cols_binarias].astype(int)

df["perfil"] = (
    "Smoke=" + df["smoke"].astype(str) +
    " | Alco=" + df["alco"].astype(str) +
    " | Active=" + df["active"].astype(str)
)

risco_por_perfil = (
```

```
    df
    .groupby("perfil")["cardio"]
    .mean()
    .sort_values()
)

risco_percentual = risco_por_perfil * 100
plt.figure(figsize=(12, 6))

bars = plt.barh(
    risco_percentual.index,
    risco_percentual.values
)

plt.xlabel("Percentual com problema cardíaco (%)")
plt.title("Risco de problema cardíaco por combinação de comportamentos")

for bar in bars:
    valor = bar.get_width()
    y = bar.get_y() + bar.get_height() / 2

    plt.text(
        valor + 0.5,
        y,
        f"{valor:.1f}%",
        va="center",
        fontsize=9
    )

plt.tight_layout()
plt.show()
```



Preparamos mais um gráfico para analisar problemas cardíacos por uma faixa etária a cada 5 anos.

```
data_frame_cardio_true = df_pradonizado[df_pradonizado["cardio"] ==
True].copy()

df = data_frame_cardio_true.copy()
df["age"] = (df["age"] // 365).astype(int)
df["age_range_5"] = (df["age"] // 5) * 5

percentual_por_faixa = (
    df["age_range_5"]
    .value_counts(normalize=True)
    .sort_index() * 100
)

maior_percentual = percentual_por_faixa.max()
faixa_pico = percentual_por_faixa.idxmax()
legenda = f"Pico: {maior_percentual:.2f}% ({faixa_pico}-{faixa_pico+4} anos)"

fig, axes = plt.subplots(1, 2, figsize=(18, 6))

axes[0].plot(percentual_por_faixa.index, percentual_por_faixa.values)

axes[0].set_xlabel("Faixa etária (anos)")
axes[0].set_ylabel("Percentual (%)")

axes[0].axhline(
    y=maior_percentual,
    linestyle="--",
    label = legenda)

axes[0].axvline(x=faixa_pico, linestyle="--")

bars = axes[1].barh(percentual_por_faixa.index, percentual_por_faixa.values)

axes[1].set_xlabel("Percentual (%)")
axes[1].axvline(x=maior_percentual,
                linestyle="--")

handles, labels = axes[0].get_legend_handles_labels()

fig.legend(
    handles,
    labels,
    frameon=False
)

fig.suptitle("Distribuição percentual por faixa etária (5 anos)", fontsize=14)
plt.tight_layout()
plt.show()
```



Treinamento

A Regressão Logística foi adotada neste trabalho por ser um método consolidado para problemas de classificação binária, especialmente em aplicações na área da saúde, onde a interpretabilidade do modelo é um fator essencial. Além de fornecer estimativas probabilísticas, o algoritmo permite o ajuste do limiar de decisão (*threshold*), o que possibilita adequar o modelo aos objetivos do contexto clínico. Os resultados obtidos indicam boa capacidade discriminativa, com acurácia em torno de 72% e valor de ROC-AUC próximo de 0,79. Observou-se que a redução do *threshold* acarretou uma leve diminuição da acurácia, compensada por um aumento relevante do recall, que se aproximou de 80%, reduzindo a ocorrência de falsos negativos. Essa característica é particularmente importante em cenários médicos, nos quais é preferível identificar potenciais pacientes em risco, ainda que isso implique em um maior número de falsos positivos, reforçando a adequação da Regressão Logística como modelo de apoio à decisão clínica.

Para treinamento inicialmente iremos separar o treino do teste

```
x = df_pradonizado.drop(["cardio"], axis = 1)
y = df_pradonizado["cardio"] # Target

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,
stratify=y, random_state=42 )
```

A padronização das variáveis foi realizada com o objetivo de melhorar a convergência do modelo e assegurar que todas as variáveis contribuíssem de forma equilibrada durante o treinamento.

```
scaler = StandardScaler()
```

```
x_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)
```

```
model_logistic = LogisticRegression(max_iter=1000)
model_logistic.fit(x_scaled, y_train)
y_prob = model_logistic.predict_proba(x_test_scaled)[:, 1]
y_pred = (y_prob >= 0.5).astype(int)

print(classification_report(y_test, y_pred))
print("Acurácia:",  accuracy_score(y_test, y_pred))
print("ROC-AUC:", roc_auc_score(y_test, y_prob))
print("Recall Score:", recall_score(y_test, y_pred))
print("Matriz de Confusão: \n", confusion_matrix(y_test, y_pred))
```

```
              precision    recall  f1-score   support

       False       0.70      0.78      0.74      6946
        True       0.75      0.67      0.70      6809

    accuracy                           0.72     13755
   macro avg       0.72      0.72      0.72     13755
weighted avg       0.72      0.72      0.72     13755

Acurácia: 0.7224282079243911
ROC-AUC: 0.78765120366893
Recall Score: 0.6670583051843149
Matriz de Confusão:
 [[5395 1551]
 [2267 4542]]
```

Os resultados mostram que o modelo alcançou uma acurácia de cerca de 72%, com boa capacidade de separação entre pacientes com e sem risco cardíaco, conforme indicado pelo ROC-AUC de aproximadamente 0,79. O recall de 67% para a classe positiva demonstra que a maior parte dos pacientes em risco foi corretamente identificada. A matriz de confusão evidencia um equilíbrio entre erros de classificação, típico de aplicações clínicas.

Para aprimorar o desempenho do modelo, foi aplicado um processo de *Grid Search* com validação cruzada, priorizando a métrica de *recall*, em consonância com o objetivo clínico do projeto. Diferentes combinações de *solver*, penalização e pesos de classe foram avaliadas.

```
pipe_logistic_regression = Pipeline([
    ("scaler", StandardScaler()),
    ("model", LogisticRegression(max_iter=1000))
])

param_grid = [
    {
        "model__solver": ["lbfgs", "newton-cg", "sag"],
        "model__penalty": ["l2"],
        "model__C": [0.1, 1, 10],
        "model__class_weight": [None, "balanced"]
    },
```

```
    {
        "model__solver": ["liblinear"],
        "model__penalty": ["l1", "l2"],
        "model__C": [0.1, 1, 10],
        "model__class_weight": [None, "balanced"]
    },
    {
        "model__solver": ["saga"],
        "model__penalty": ["l1", "l2", "elasticnet"],
        "model__l1_ratio": [0.5],
        "model__C": [0.1, 1, 10],
        "model__class_weight": [None, "balanced"]
    }
]

grid = GridSearchCV(
    estimator=pipe_logistic_regression,
    param_grid=param_grid,
    scoring="recall", # priorizando recall
    cv=5,
    n_jobs=-1,
    verbose=2
)

grid.fit(X_train, y_train)
best_model = grid.best_estimator_

print("Melhor Modelo", best_model)
```

```
Fitting 5 folds for each of 48 candidates, totalling 240 fits
Melhor Modelo Pipeline(steps=[('scaler', StandardScaler()),
               ('model',
                LogisticRegression(C=1, class_weight='balanced', max_iter=1000,
                                   solver='newton-cg'))])
```

Encontrando o Threshold ideal

```
fpr, tpr, thresholds = roc_curve(y_test, y_prob)
best_idx = np.argmax(tpr - fpr)
```

Melhor threshold em geral

```
best_threshold = thresholds[best_idx]
```

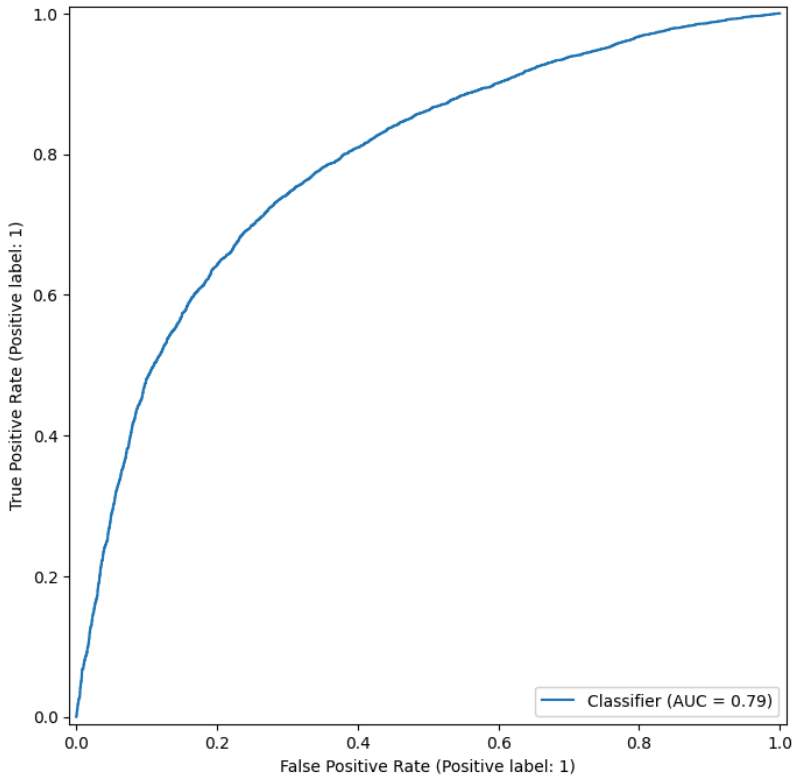
Melhor threshold para garantir um recall acima de 80%

```
best_recall_threshold = thresholds[tpr >= 0.80][0]

print("Threshold ideal:", best_threshold)
print("Threshold ideal para garantir um recall acima de 80%:",
best_recall_threshold)
```

```
Threshold ideal: 0.48445910721289326
Threshold ideal para garantir um recall acima de 80%: 0.399676365687596
```

Criando novo pipeline com parâmetros e threshold ideais

```
pipe_logistic_regression_best = Pipeline([
    ("scaler", StandardScaler()),
    ('model',LogisticRegression(C=1, class_weight='balanced',
max_iter=1000,solver='newton-cg'))
])
pipe_logistic_regression_best.fit(x_train, y_train)
y_prob = pipe_logistic_regression_best.predict_proba(x_test)[:, 1]
RocCurveDisplay.from_predictions(
    y_test, y_prob / 100
)
plt.show()
```



A curva ROC-AUC indica que, ao comparar aleatoriamente um paciente doente com um paciente saudável, o modelo atribui uma probabilidade maior ao paciente doente em aproximadamente 79% das vezes.

Testando os valores de threshold identificados anteriormente, utilizando o melhor threshold, o valor do recall está abaixo do aceitável

```
y_pred = (y_prob >= best_threshold).astype(int)

print(classification_report(y_test, y_pred))
print("Acurácia:",  accuracy_score(y_test, y_pred))
print("Recall:", recall_score(y_test, y_pred))
print("Matriz de Confusão: \n", confusion_matrix(y_test, y_pred))
```

Utilizando o melhor threshold para garantir o recall acima de 80%. Dessa forma, os casos falsos negativos irão diminuir, entretanto teremos menos casos falsos positivos

```
y_pred = (y_prob >= best_recall_threshold).astype(int)
```

```
print(classification_report(y_test, y_pred))
print("Acurácia:",  accuracy_score(y_test, y_pred))
print("Recall Score:", recall_score(y_test, y_pred))
print("Matriz de Confusão: \n", confusion_matrix(y_test, y_pred))
```

```
              precision    recall  f1-score   support

       False       0.72      0.75      0.73      6946
        True       0.73      0.69      0.71      6809

    accuracy                           0.72     13755
   macro avg       0.73      0.72      0.72     13755
weighted avg       0.73      0.72      0.72     13755

Acurácia: 0.7245365321701199
Recall: 0.6949625495667499
Matriz de Confusão:
 [[5234 1712]
 [2077 4732]]
```

Apesar de utilizar um valor abaixo do ideal para o threshold, o modelo perdeu ~2% de acurácia. Entretanto, quando pensamos em um contexto médico/hospitalar, é melhor termos um número de falsos positivos (pacientes que não tem problemas cardíacos, porém foram identificados com tal), do que correr o risco de deixarmos algum paciente doente passar sem identificação.

Utilizando o threshold para priorizar o valor de recall, o modelo apresentou uma acurácia de ~71%, mantendo o valor de recall em ~80%

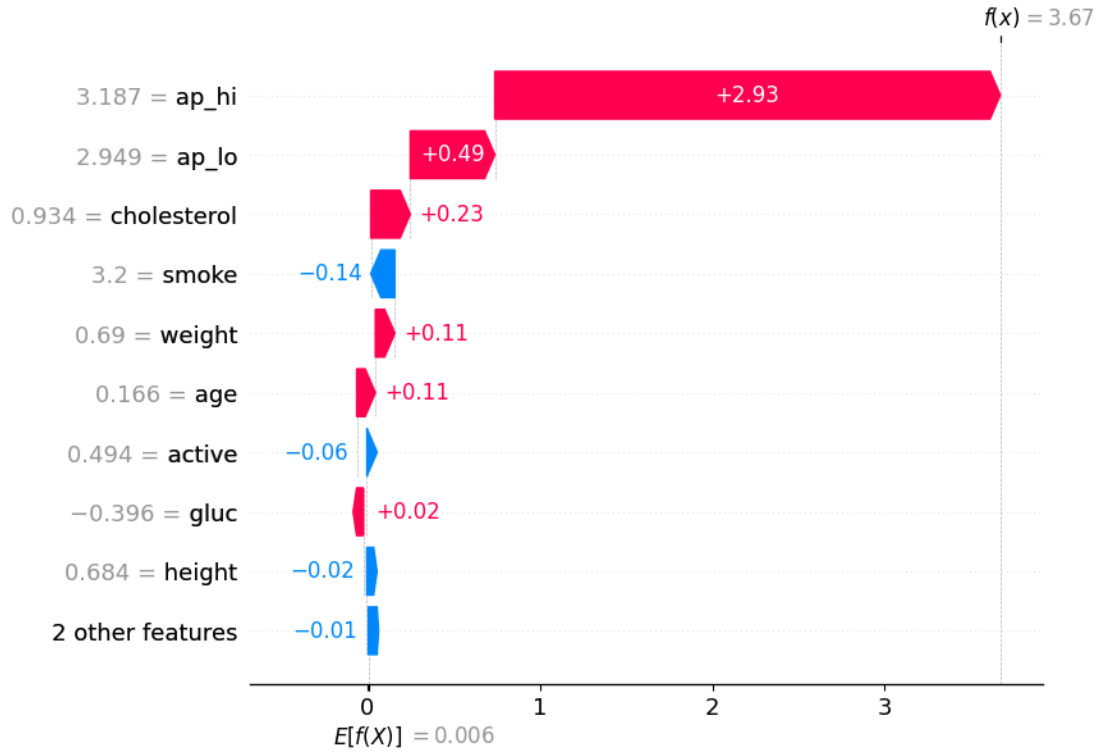
A utilização da biblioteca SHAP permitiu analisar a contribuição individual de cada variável para a decisão do modelo, aumentando a transparência do processo preditivo. A análise de um caso específico demonstrou como determinadas características do paciente influenciaram diretamente a alta probabilidade estimada de ocorrência de problema cardíaco.

```
X_train_scaled =
pipe_logistic_regression_best.named_steps["scaler"].transform(x_train)
X_test_scaled =
pipe_logistic_regression_best.named_steps["scaler"].transform(x_test)

feature_names = x.columns

explainer = shap.LinearExplainer(
    pipe_logistic_regression_best.named_steps["model"],
    X_train_scaled,
```

```
    feature_names=feature_names
)

shap_values = explainer(X_test_scaled)
```

Selecionando um caso aleatório dentro dos casos separados para teste

```
i = np.random.randint(0, X_test_scaled.shape[0])

base_log_odds = explainer.expected_value
patient_log_odds = base_log_odds + shap_values[i].values.sum()

prob = 1 / (1 + np.exp(-patient_log_odds))

shap.plots.waterfall(shap_values[i])
print(f"Probabilidade do paciente ter problema cardiáco: {prob}")
```




```
Probabilidade do paciente ter problema cardiáco: 0.9751555964892844
```

Conclusão

Os resultados obtidos ao longo deste estudo demonstram que a Regressão Logística é uma abordagem adequada para a avaliação do risco de doenças cardíacas, apresentando bom desempenho preditivo e elevada interpretabilidade. O modelo alcançou uma acurácia em torno de 72% e um valor de ROC-AUC próximo de 0,79, indicando capacidade satisfatória de

discriminação entre pacientes com e sem risco cardíaco. A análise do ajuste do threshold evidenciou um trade-off controlado entre acurácia e recall, sendo possível priorizar a identificação de pacientes em risco, característica essencial em contextos clínicos. Além disso, o uso de técnicas de interpretabilidade, como SHAP, reforçou a confiabilidade do modelo ao permitir compreender os fatores que influenciam suas decisões. Dessa forma, o modelo desenvolvido mostra-se apropriado como ferramenta de apoio à decisão clínica, podendo auxiliar profissionais de saúde na triagem e no acompanhamento de pacientes.