Análise do Dataset Heart Disease UCI

Diego Rodrigues Medeiros [1], Júlia Neves Guardiani [2]

[1] Instituto Metrópole Digital – Universidade Federal do Rio Grande do Norte (UFRN) Campus Universitário – 59.078-970 – Natal - Rio Grande do Norte – RN – Brazil

diego25rn@gmail.com,julia.guardiani.089@ufrn.edu.br

1. Introdução

A base de dados escolhida para a realização do trabalho foi “Heart Disease UCI”, relacionado a doenças cardíacas. Esse csv contém 76 atributos, mas todos os experimentos publicados referem-se ao uso de um subconjunto de 14 deles. Com o intuito de estudar sobre quais grupos de pessoas são mais propícias a desenvolverem doenças cardíacas, criou-se uma nota para as determinadas situações, sendo 0 (sem presença) a 4 (Alta probabilidade). Entretanto neste trabalho iremos lidar somente com 0(sem presença) e 1(probabilidade existente)

A base foi retirada do *kaggle* (rede social e comunidade de ciência de dados) e tem classificação ouro pela quantidade de pessoas trabalhando com ela e usabilidade 7.6 devido aos resultados mais comuns dos modelos de aprendizagem de máquina simples implementados. Na plataforma do *kaggle* é possível encontrar aproximadamente 1800 trabalhos envolvendo a base, e, em sua grande maioria, são códigos *python* voltados para modelos de classificação simples e ou análise exploratória dos dados.

Neste trabalho iremos utilizar a base de dados para treinar modelos de aprendizado supervisionado no intuito de classificar se uma pessoa tem chance alta ou baixa de contrair uma doença cardíaca com base em suas características físicas. Por mais que essa base seja bastante trabalhada e tenha um número relativamente baixo de características para a predição, o estudo é importante dado o seu contexto geral voltado para a saúde.

Quanto mais estudos relacionados à previsão de doenças cardíacas, maiores são as chances de melhorar os tratamentos preventivos no futuro, o que seria deveras importante para o país dada a alta taxa de mortalidade no Brasil relacionada a doenças cardíacas. Taxa essa que pode ser observada de forma estimada no site “Cardiômetro” desenvolvido pela sociedade brasileira de cardiologia.

Neste trabalho iremos lidar com modelos supervisionados sendo eles individuais ou comitês de classificadores para desenvolvermos modelos de aprendizado de máquina robustos e com uma precisão desejável 80% ou superior.

2. Descrição da Base de Dados

A base de dados como já dito anteriormente diz respeito a características laboratoriais de um indivíduo que podem ocasionar o desenvolvimento de doenças cardíacas.

A base foi retirada do kaggle[1], possui 14 colunas e 303 registros. Neste trabalho iremos descrever o significado de coluna bem como suas características para que nos próximos tópicos, possamos justificar os tipos de análise escolhidos para explorar os dados.

2.1. Descrição dos dados e seus significados

Abaixo construímos uma tabela com os dados e seu significado no contexto trabalhado. Os dados numéricos foram adicionados com suas respectivas médias e desvios padrões.

Sigla	**Descrição**
Age[**Numeric**]	Idade do paciente média: 54.36 Desvio padrão: 9.08
sex [**Categorical**]	Gênero do paciente: 1: Masculino 0: Feminino
exng[**Categorical**]	Angina induzida por exercício
caa[**Categorical**]	Número de veias principais coloridas por fluoroscopia
cp[**Categorical**]	Tipo de dor sentida pelo paciente: 0: Assintomático 1: Angina típica 2: Angina atípica 3: Dor diferente angina
trtbps [**Numeric**]	pressão sanguínea em descanso média: 131.62 desvio padrão: 17.54
chol [**Numeric**]	colesterol em mg/dl média: 246.26 desvio padrão:51.83
fbs [**Categorical**]	Açúcar no sangue em jejum > 120 mg/dl: 1 : Verdadeiro 0 : Falso
restecg [**Categorical**]	Resultado do eletrocardiograma em repouso 0: Normal 1: Possui anormalidade nas ondas 2: Provável hipertrofia ventricular esquerda
thalachh[**Numeric**]	Frequência cardíaca máxima alcançada no

	exame. média: 149.65 desvio padrão: 22.91
Oldpeak[**Numeric**]	depressão da escápula induzida pelo exercício, calculado em relação ao descanso média: 1.04 desvio padrão: 1.16
slp [**Categorical**]	inclinação escapular no pico do exercício 0: inclinação ascendente 1: plano 2: inclinação descendente
Thall [**Categorical**]	Taxa de Thalassemia 0: Erro(2 dados faltantes mantidos) 1: Normal 2: Defeito fixo 3: Defeito reversível
output**[Label]**	saída: 0: Pouca chance de um ataque cardíaco 1: Chance alta de um ataque cardíaco.

2.2 Análise exploratória

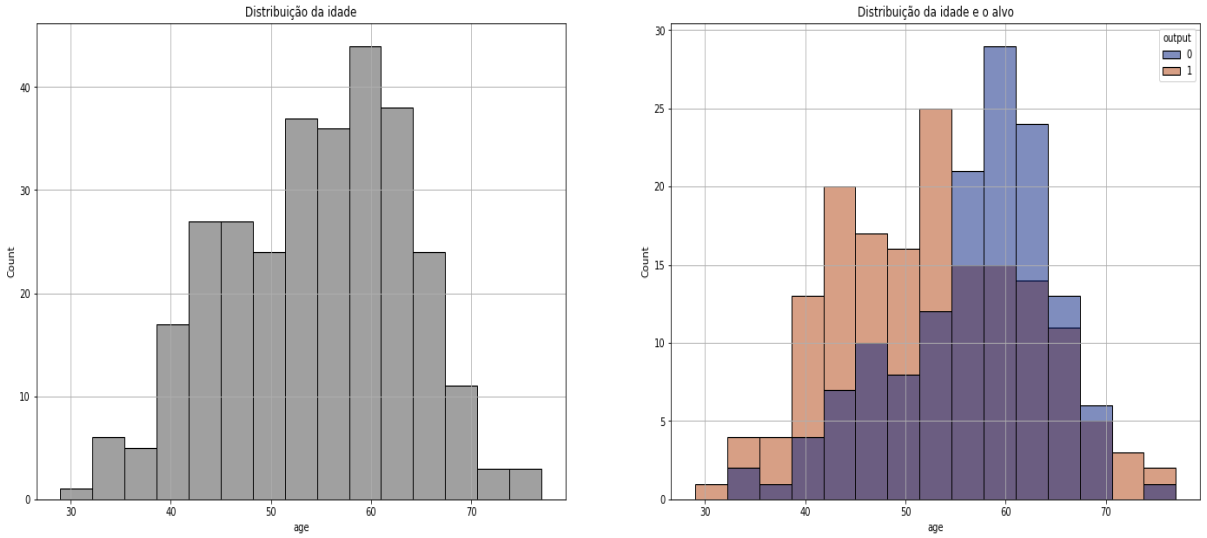
Dadas as características dos dados que foram exibidos no tópico anterior foi decidido que a análise exploratória seria feita com duas abordagens diferentes com base no tipo de coluna trabalhada.

Para isso iremos considerar dados numéricos discretos como categóricos para padronizar o tipo de visualização utilizada. Dito isso as abordagens serão: Para dados numéricos serão utilizados histogramas e “boxplots” para visualizarmos como está as distribuição dos dados em um contexto geral e em relação à classe alvo ‘output’ para analisarmos como a distribuição pode ser alterada em relação à saída; Para dados categóricos serão feitos ‘k+1’ gráficos de barra sendo ‘k’ o número de categorias presentes na tabela, ou seja estudaremos primeiro como está a proporção dos dados no contexto geral da classe e em seguida detalharemos a distribuição de cada categoria em detrimento da saída.

Esse tópico será dividido em dois blocos, o primeiro fará a análise gráfica das colunas e o segundo bloco será feita uma análise gráfica no intuito de analisar a correlação das colunas trabalhadas. Todos os gráficos foram gerados utilizando a biblioteca *Seaborn* do *python*.

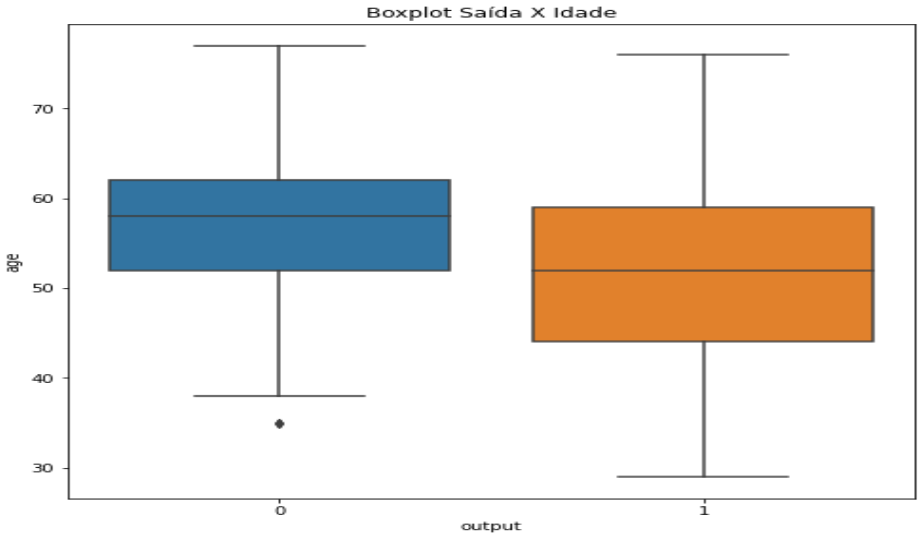
2.2.1 Análise gráfica das colunas
2.2.1.1 Idade(age)

Figura 1- Distribuição das idades



Fonte: Autoria própria

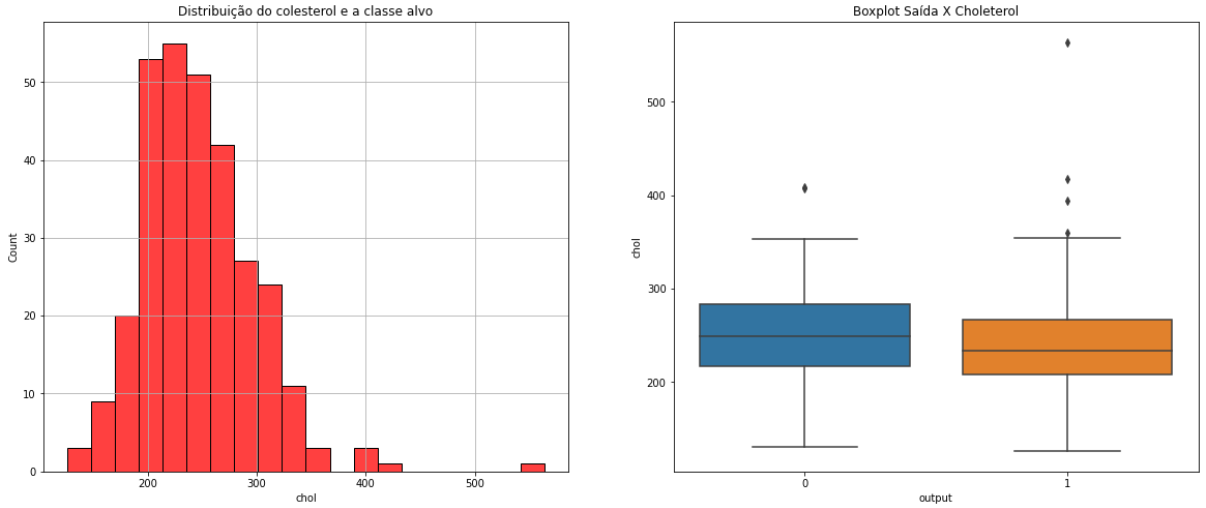
Figura 2- Distribuição das idades Boxplot



Fonte: Autoria própria

2.2.1.2 Colesterol(chol)

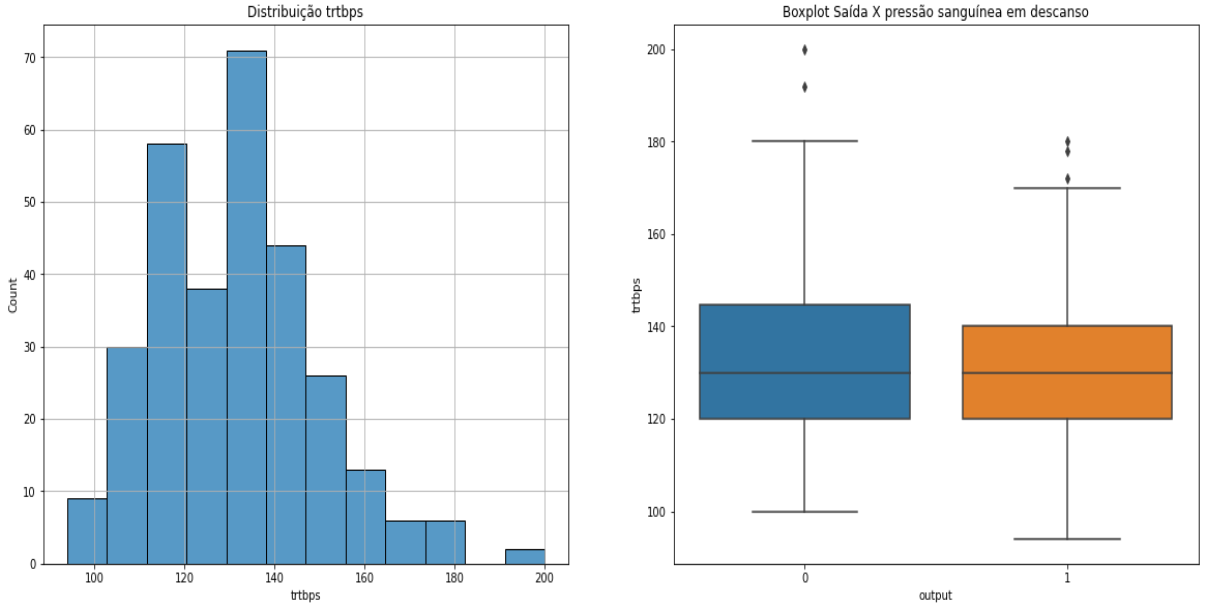
Figura 3- Distribuição do colesterol histograma e Boxplot



Fonte: Autoria própria

2.2.1.3 Pressão sanguínea em descanso

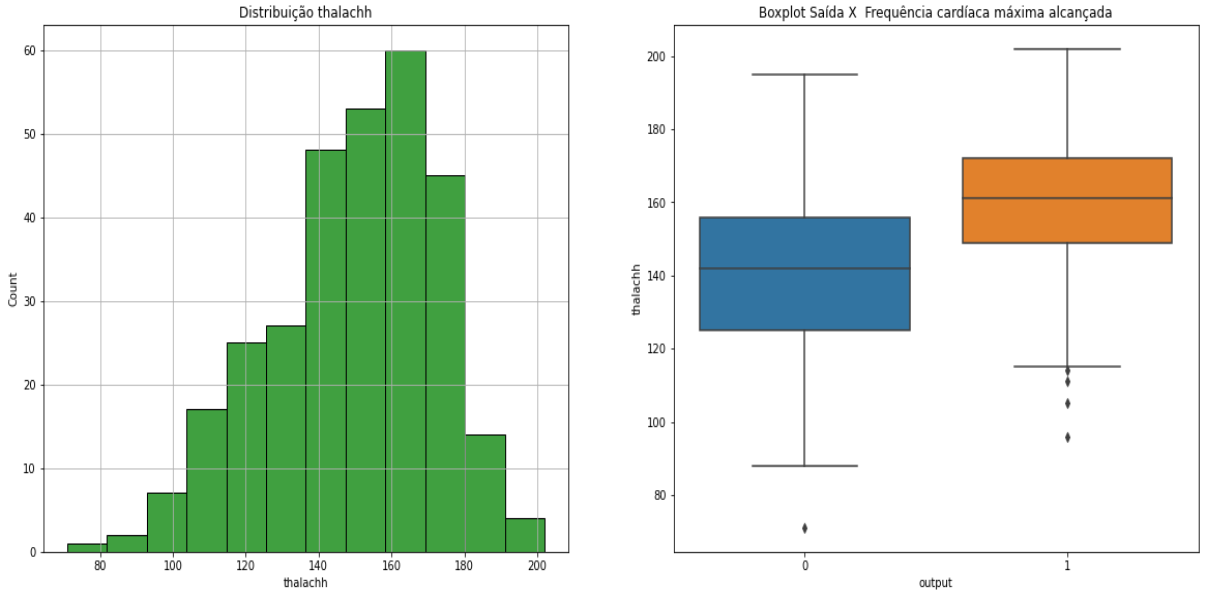
Figura 4 - Distribuição da pressão sanguínea no descanso



Fonte: Autoria própria

2.2.1.4 Frequência cardíaca máxima alcançada

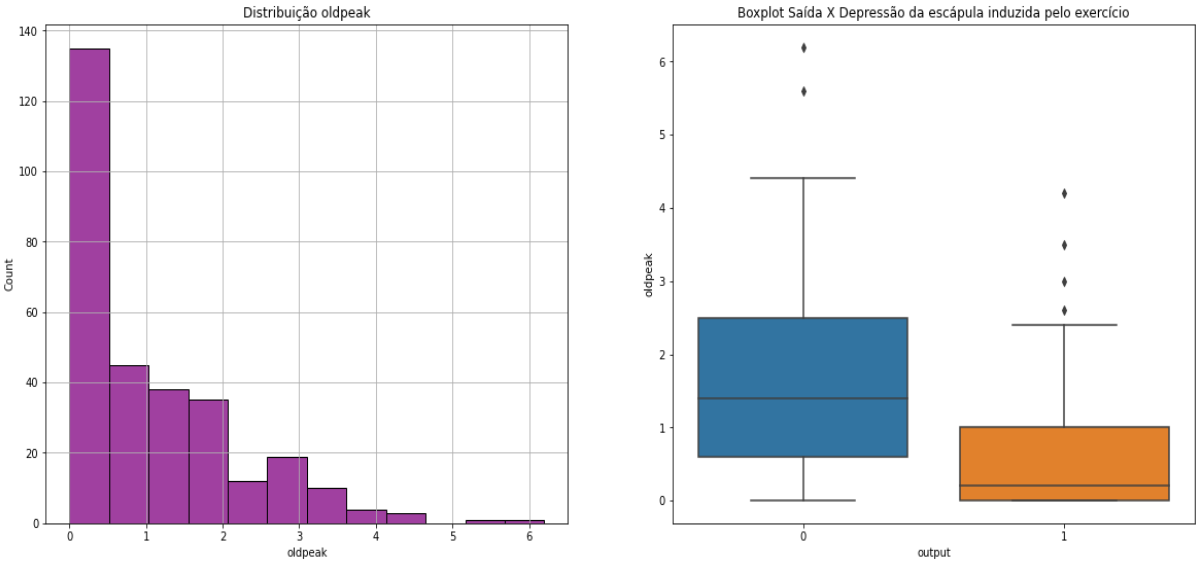
Figura 5- Distribuição da frequência cardíaca máxima alcançada



Fonte: Autoria própria

2.2.1.5 Depressão da escápula induzida por exercício

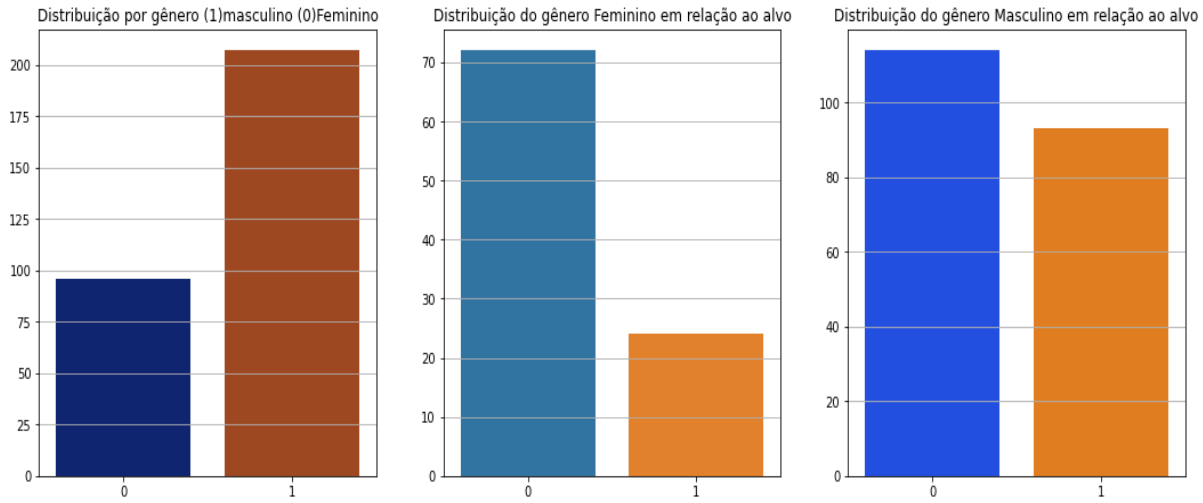
Figura 6- Distribuição da depressão da escápula induzida por exercício



Fonte: Autoria própria

2.2.1.6 Gênero

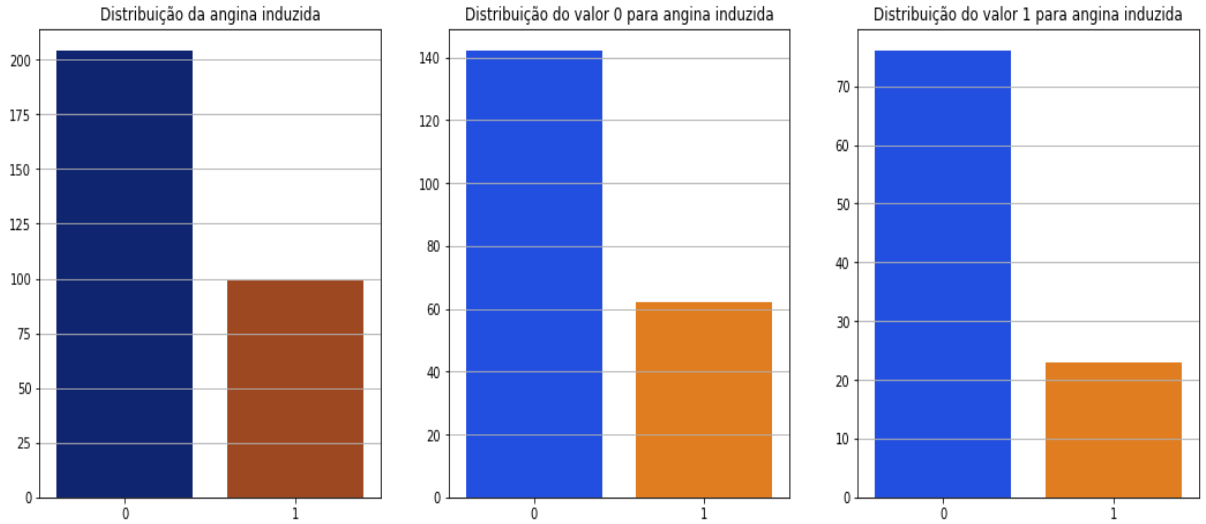
Figura 7- Proporção do gênero



Fonte: Autoria própria

2.2.1.7 Angina induzida por exercício

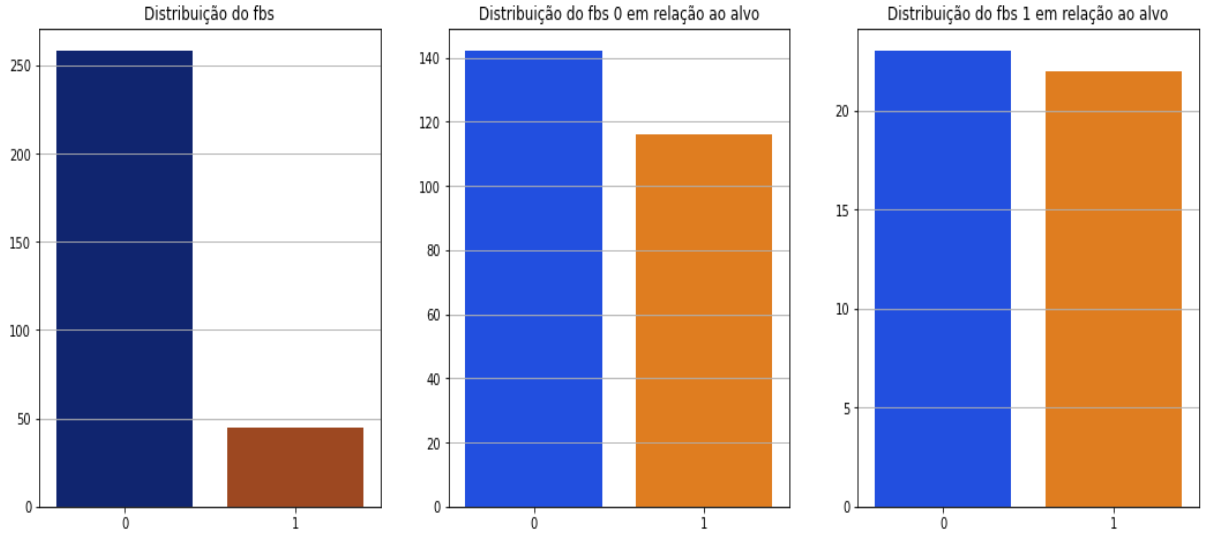
Figura 8- Proporção da angina induzida por exercício



Fonte: Autoria própria

2.2.1.8 Taxa de açúcar no sangue > 120 mg/dl (fbs)

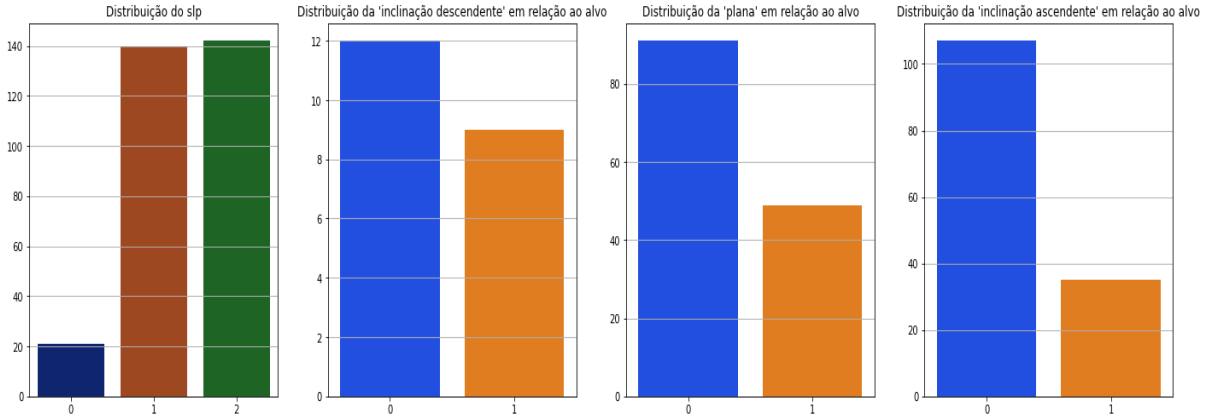
Figura 9- Proporção do fbs



Fonte: Autoria própria

2.2.1.9 Inclinação da escápula(slp)

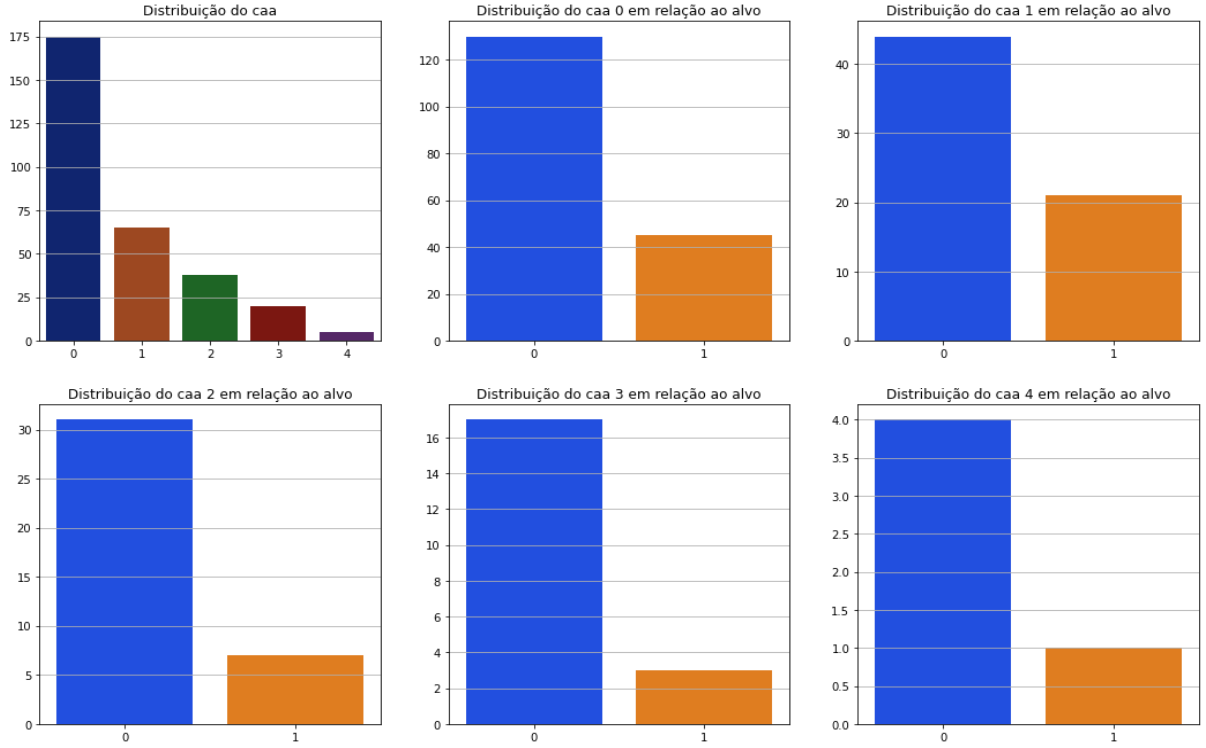
Figura 10- Proporção da slp



Fonte: Autoria própria

2.2.1.10 Número de veias principais coloridas por fluoroscopia(caa)

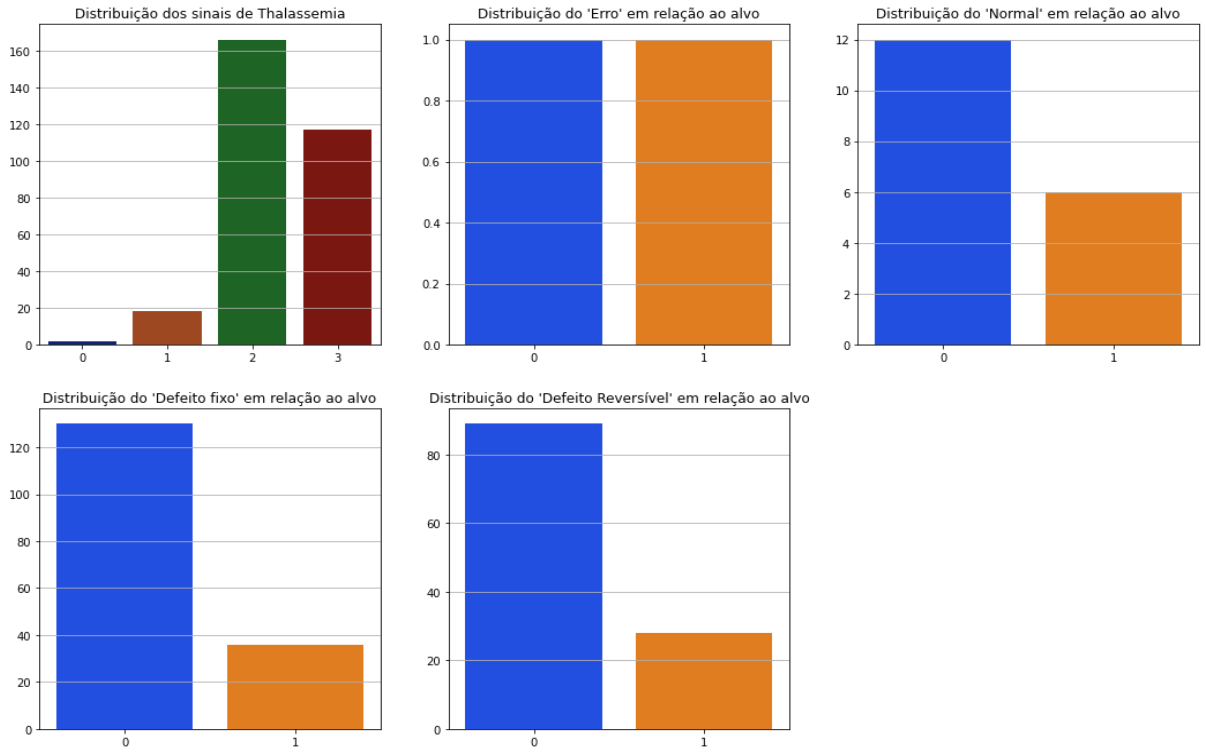
Figura 11- Proporção do caa



Fonte: Autoria própria

2.2.1.11 Taxa de Thalassemia

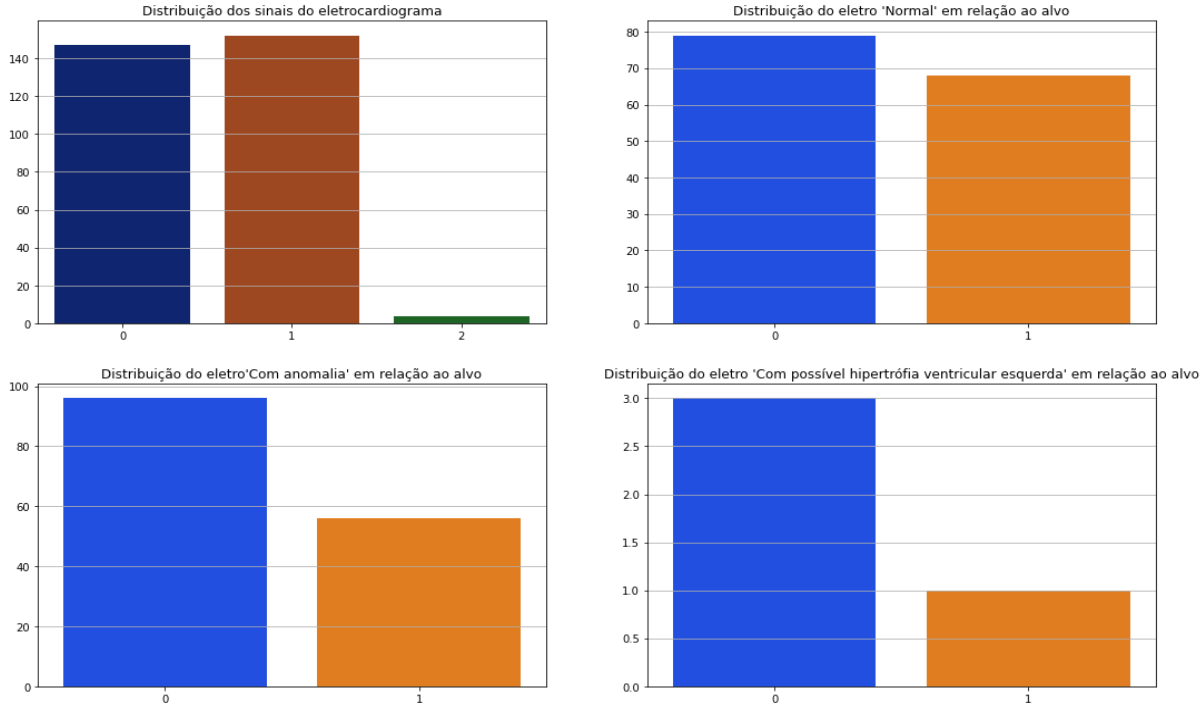
Figura 12- Proporção da Taxa de thalassemia



Fonte: Autoria própria

2.2.1.12 Sinais no eletrocardiograma

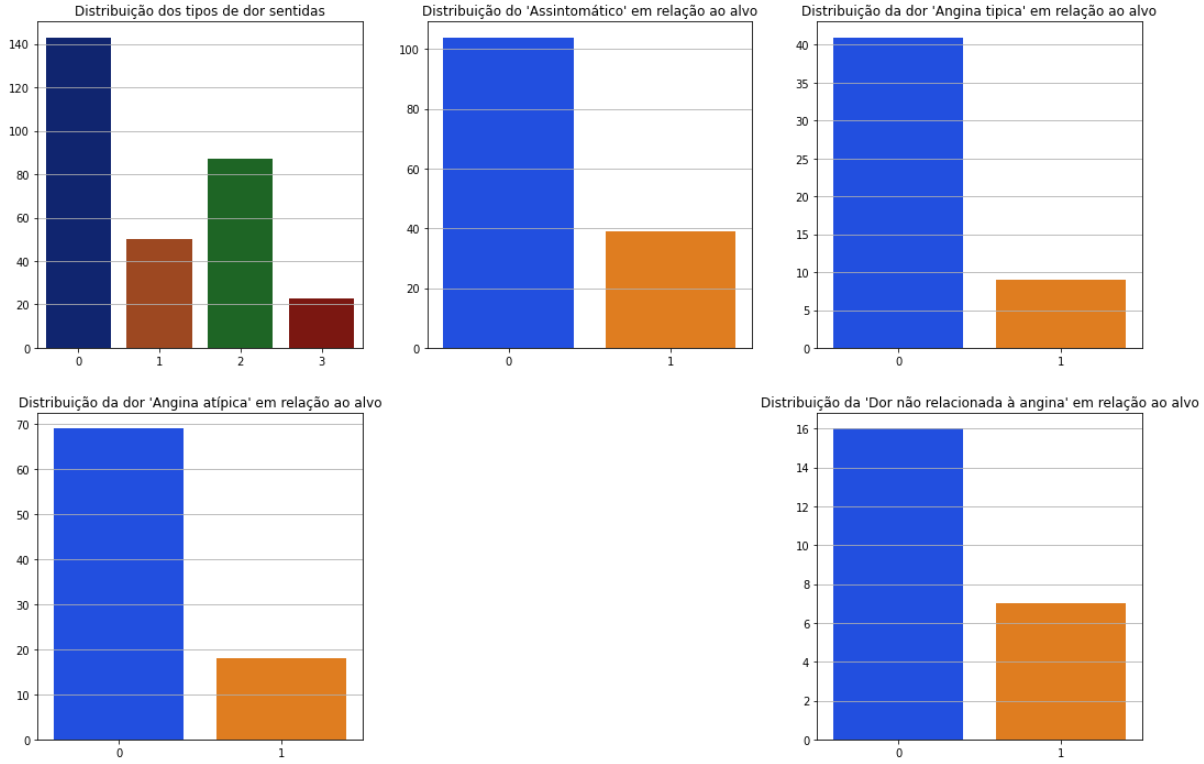
Figura 13- Proporção do sinais no eletro



Fonte: Autoria própria

2.2.1.13 Tipo de dor sentida

Figura 14- Proporção do tipo da dor

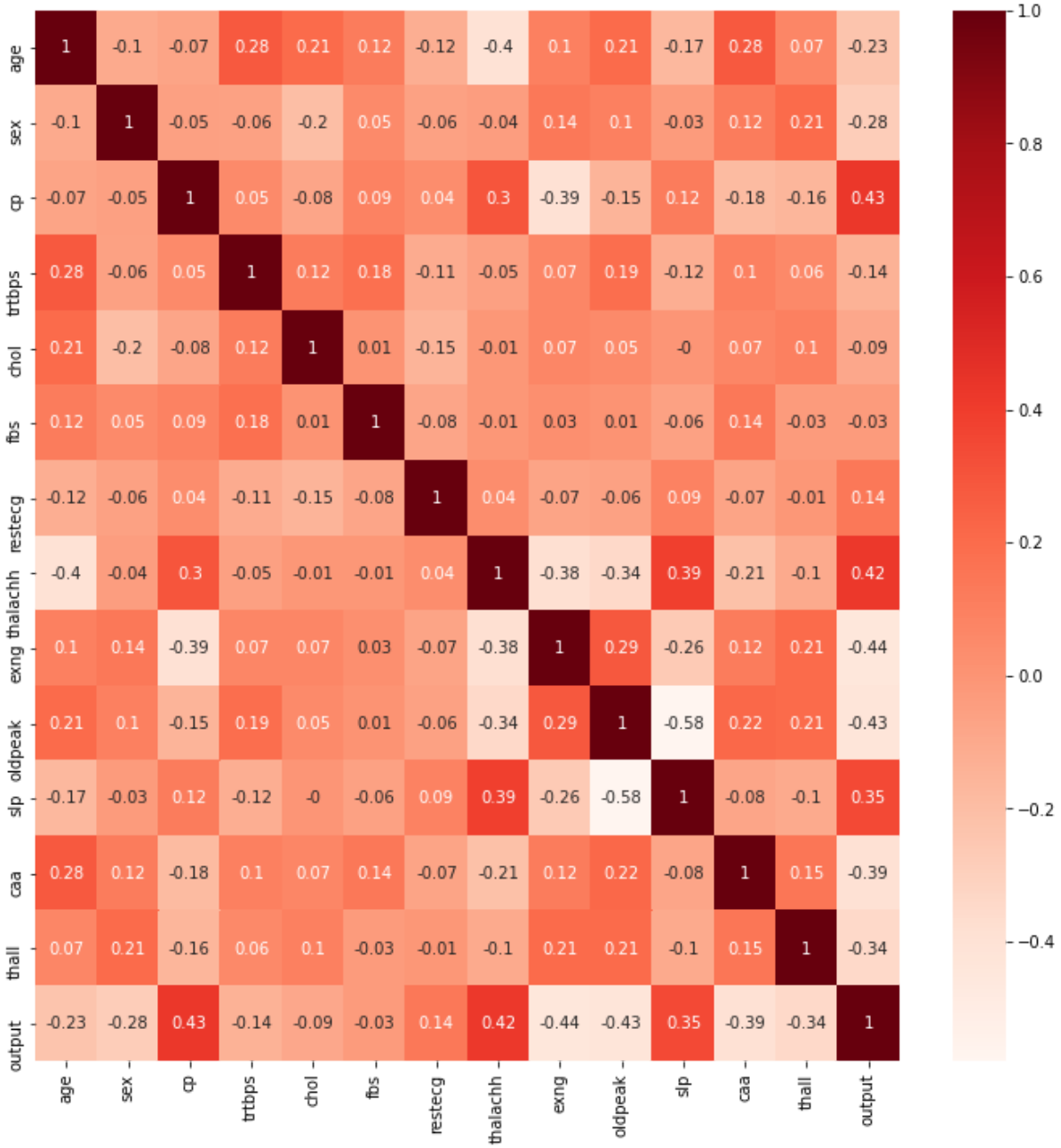


Fonte: Autoria própria

2.2.2. Análise de correlação

2.2.2.1 Mapa de calor

Figura 15 - Mapa de calor da matriz de correlação



3.2.2 Scatterplot a pares

Figura 16 - Pairplot para análise de correlação a pares

Com essas visualizações, tanto a distribuição dos valores contínuos quanto os categóricos podem ser analisados em relação à variável de saída. Por exemplo, ao analisar os histogramas da idade podemos perceber que pessoas com maior chance de ataque cardíaco se concentram em uma faixa de idade diferente da outra classe presente no label de saída. Desta forma podemos ter insights de quais colunas tem maior variação de comportamento com relação à classe alvo.

Os gráficos de correlação, ajudam a identificar quais são as colunas mais relacionadas com a saída e também quais são as colunas que possuem comportamento parecido em relação à saída.

Os insights iniciais que foi possível retirar dos gráficos são: a forma como a idade parece influenciar no resultado que estamos classificando, já que ao exibir os valores positivos e negativos da classe alvo no mesmo gráfico, observamos uma mudança na distribuição dos dados na visualização, pessoas de idade mediana parecem ter mais chances de desenvolver uma doença cardíaca do que pessoas de idade mais avançada; Além da idade, os valores de pico dos batimentos cardíacos são relativamente maiores em pessoas propensas a desenvolver problemas cardíacos, e a curvatura da escápula também possui uma distribuição de valor mais diferenciada em detrimento com a classe alvo.

Os valores categóricos também possibilitam algumas hipóteses. O gênero possui uma quantidade proporcional maior de homens com tendências a desenvolver doenças no gráfico de barras, a inclinação da escápula de acordo com o valor numérico também possui proporções crescentes em acordo com as categorias observadas.

Os gráficos de correlação não trouxeram grandes informações para a análise dos dados de fato. Grande maiorias das colunas utilizadas no estudo possuem correlação baixa e portanto podemos concluir que inicialmente, não seria interessante fazer nenhum redução das dimensionalidades dos dados utilizados

3. Pré-Processamento

Antes do pré-processamento foi utilizada a biblioteca de autoML “pycaret” do python 3.6+ para visualizar se a normalização por z-score padrão e os métodos de binarização com “one hot encoding” seriam suficientes para que o trabalho alcançasse um bom resultado com relação aos modelos.

Os melhores modelos encontrados para a base de dados utilizada foram a regressão logística e a floresta aleatória. Contudo, os modelos que serão utilizados para a experimentação presente neste trabalho apresentaram um desempenho aquém do desejado inicialmente na proposta inicial.

Entretanto, a normalização po*r z-score* e o *one hot encoding* passados como parâmetro de experimentação para a biblioteca de autoML possibilitou a escolha desses métodos para o pré-processamento dos dados, já que o resultado do melhor modelo teve um desempenho acima dos 80% mínimos desejados inicialmente para os resultados deste trabalho.

Figura 17 - AutoML

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
rf	Random Forest Classifier	0.8532	0.9050	0.8644	0.8756	0.8664	0.7029	0.7098	0.463
et	Extra Trees Classifier	0.8485	0.8881	0.8644	0.8677	0.8641	0.6926	0.6972	0.455
ridge	Ridge Classifier	0.8437	0.0000	0.8985	0.8412	0.8642	0.6790	0.6936	0.014
lda	Linear Discriminant Analysis	0.8437	0.9044	0.8985	0.8412	0.8642	0.6790	0.6936	0.016
lr	Logistic Regression	0.8390	0.9076	0.8652	0.8595	0.8514	0.6749	0.6910	0.493
ada	Ada Boost Classifier	0.8154	0.8692	0.8561	0.8290	0.8365	0.6232	0.6350	0.095
gbc	Gradient Boosting Classifier	0.7922	0.8939	0.8295	0.8074	0.8126	0.5760	0.5857	0.086
lightgbm	Light Gradient Boosting Machine	0.7872	0.8814	0.8220	0.8014	0.8099	0.5672	0.5705	0.075
nb	Naive Bayes	0.7589	0.8818	0.6205	0.9329	0.7275	0.5350	0.5801	0.015
dt	Decision Tree Classifier	0.7372	0.7318	0.7826	0.7569	0.7659	0.4663	0.4735	0.015
qda	Quadratic Discriminant Analysis	0.6450	0.6391	0.6992	0.6874	0.6828	0.2772	0.2848	0.018
knn	K Neighbors Classifier	0.6325	0.6784	0.7303	0.6529	0.6858	0.2432	0.2521	0.118
svm	SVM - Linear Kernel	0.5762	0.0000	0.5500	0.4991	0.4867	0.1447	0.1612	0.014

✓ 22s conclusão: 23:59

3.1 One hot encoding

Como os atributos gerados no “dataset” são divididos em dois grupos: numéricos e categóricos. Estabeleceu-se que o tipo numérico será tratado com a normalização padrão para conseguir uma leitura melhor dos dados.

Já os atributos categóricos serão tratados com o one hot encoding para a binarização. As colunas alteradas são: *restecg, cp, slp*, *thall* e *caa*. Algumas dessa colunas passaram por one label encoding dando nomes para os valores numérico sem valor ordinal. Por exemplo, o restecg passou pela tranformação 0->normal, 1-> possui_anormalidade, 2 -> provável_hipertrofia; assim, quando utilizarmos o one hot encoding, as novas colunas terão um significado mais claro com relação ao problema. Algumas colunas categóricas não passaram pela renomeação por não haverem uma descrição clara do significado numérico dentro da base de dados, sendo assim, colunas como “slp” e “thall” deram origem a novas colunas com o formato “slp_n” e “thall_n” sendo n um número representante na categoria.

Figura 18 a- classes nomeadas.

restecg_Normal	restecg_Possui_anormalidade	restecg_Provável_hipertrofia	cp_Angina_atipica
1	0	0	0
0	1	0	1
1	0	0	0
0	1	0	0
0	1	0	0

Figura 18 b- classes nomeadas.

cp_Angina_tipica	cp_Assintomatico	cp_Não_Angina
0	0	1
0	0	0
1	0	0
1	0	0
0	1	0

Figura 19- Dados originais tratados numericamente.

slp_0	slp_1	slp_2	thall_0	thall_1	thall_2	thall_3	caa_0	caa_1	caa_2	caa_3
1	0	0	0	1	0	0	1	0	0	0
1	0	0	0	0	1	0	1	0	0	0
0	0	1	0	0	1	0	1	0	0	0
0	0	1	0	0	1	0	1	0	0	0
0	0	1	0	0	1	0	1	0	0	0

3.2. Normalização

As variáveis numéricas presentes na base de dados utilizadas tiveram que passar por uma normalização. A normalização escolhida foi a normalização “padrão” ou por z-score. A normalização foi escolhida por ter sido observada a distribuição normal dos dados numéricos e também por ter sido ela a normalização utilizada durante a experimentação com o AutoML.

A normalização se faz necessária para diminuir a diferença de escala entre os diversos campos presentes dentro da base de dados utilizada, já que os modelos que serão utilizados possuem sensibilidade a diferenças bruscas de escalas. Modelos como o knn e a MLP podem demorar muito para convergir devido a escalas mal configuradas.

Figura 20- Dados originais normalizados.

	age	sex	trtbps	chol	fbs	thalachh	exng	oldpeak	slp_0	slp_1	slp_2	thall_0	thall_1	thall_2
count	303.00	303.00	303.00	303.00	303.00	303.00	303.00	303.00	303.00	303.00	303.00	303.00	303.00	303.00
mean	0.00	0.68	-0.00	-0.00	0.15	-0.00	0.33	-0.00	0.07	0.46	0.47	0.01	0.06	0.55
std	1.00	0.47	1.00	1.00	0.36	1.00	0.47	1.00	0.25	0.50	0.50	0.08	0.24	0.50
min	-2.80	0.00	-2.15	-2.32	0.00	-3.44	0.00	-0.90	0.00	0.00	0.00	0.00	0.00	0.00
25%	-0.76	0.00	-0.66	-0.68	0.00	-0.71	0.00	-0.90	0.00	0.00	0.00	0.00	0.00	0.00
50%	0.07	1.00	-0.09	-0.12	0.00	0.15	0.00	-0.21	0.00	0.00	0.00	0.00	0.00	1.00
75%	0.73	1.00	0.48	0.55	0.00	0.72	1.00	0.48	0.00	1.00	1.00	0.00	0.00	1.00
max	2.50	1.00	3.91	6.14	1.00	2.29	1.00	4.45	1.00	1.00	1.00	1.00	1.00	1.00

É importante observar a característica desse tipo de normalização. O z-score faz com que a média dos valores normalizados vá para 0 e o desvio padrão seja fixado o mais próximo possível de 1. Essas características tornam o problema mais escalável para volumes de dados maiores por reduzir a complexidade dos cálculos.

4. Experimentos

Neste tópico serão tratados dos experimentos com os modelos de aprendizagem estudados no intuito de encontrar qual seria o melhor classificador para o problema proposto neste trabalho.

Vale salientar que no tópico anterior há um pequeno trecho introduzindo o conceito de autoML e exibindo uma análise preliminar de quais seriam os modelos mais adequados para a base trabalhada, incluindo modelos de comitês de classificação como o boost.

Inicialmente iremos analisar como se comportam os modelos de maneira isolada e no tópico 4.2 iremos tratar dos ganhos e ou perdas de desempenho que podem ser obtidos com a aplicação de comitês de classificadores.

4.1 Modelos de classificação supervisionada

Nesse tópico serão mostrados os experimentos feitos com classificadores individuais. Os classificadores utilizados foram o knn, árvore de decisão, naive bayes e a mlp.

Para o knn foram alterados os valores de vizinhos para que fosse possível observar como o modelo se comportava dependendo do número de vizinhos que fossem passados como parâmetro. O knn, a árvore de decisão e a mlp foram feitas com valores de amostragem padrão de 30,50 e 70% para fazer o teste estatístico posteriormente nesta seção .

A mlp foi passada com os valores padrões do scikit learn variando a quantidade camadas escondidas de 20 a 24 camadas para testarmos como o modelo performa com relação à acurácia. Os demais parâmetros foram:
{`learning_rate='constant', activation='identity', solver='sgd'`}

Para a árvore de decisão foi variada a profundidade funcionando como uma poda do modelo para testar o quanto esse parâmetro influencia na performance da acurácia.

Para o naive bayes não foi pensado em formas de modificar os parâmetros do modelo já que o Naive bayes Gaussiano é o único que aceita valores numéricos dentro do scikit-learn e ao avaliarmos outros modelos fora deste relatório a performance dos dados discretizados não compensam em termos da acurácia.

4.1.1 Antes do processamento

k-NN						
Base	**Treinamento/Teste**	**1k**	**2k**	**3k**	**4k**	**5k**
Metodologias		**Acc**	**Acc**	**Acc**	**Acc**	**Acc**
Base Original	10-fold CV	0.5945	0.5679	0.6305	0.6072	0.6535
	70/30	0.5494	0.5604	0.6593	0.6593	0.6703
	80/20	0.5574	0.5738	0.6885	0.5902	0.5737
	90/10	0.5902	0.5738	0.6885	0.6066	0.5574
Amostra 70%	10-fold CV	0.5788	0.5597	0.6647	0.6173	0.6177
	70/30	0.6406	0.6406	0.5625	0.6406	0.5468
	80/20	0.5349	0.5581	0.6279	0.6744	0.5814
	90/10	0.5349	0.6512	0.6279	0.6046	0.5581
Amostra 50%	10-fold CV	0.6271	0.6612	0.69	0.6871	0.6879
	70/30	0.6739	0.5869	0.6521	0.6522	0.6956
	80/20	0.6129	0.5483	0.5161	0.6452	0.5806
	90/10	0.5161	0.7097	0.5806	0.7097	0.6774
Amostra 30%	10-fold CV	0.4833	0.5944	0.4867	0.5199	0.5644
	70/30	0.3928	0.4642	0.6071	0.5714	0.4643
	80/20	0.5263	0.5789	0.5789	0.6316	0.4737
	90/10	0.3684	0.4210	0.5263	0.6316	0.5789

Árvore de Decisão						
Base	**Treinamento/Teste**	**Prof. 1**	**Prof. 2**	**Prof. 3**	**Prof. 4**	**Prof. 5**
Metodologias		**Acc**	**Acc**	**Acc**	**Acc**	**Acc**
Base Original	10-fold CV	0.7353	0.7390	0.7685	0.7616	0.7751
	70/30	0.7473	0.7143	0.7363	0.8241	0.7252
	80/20	0.7705	0.8033	0.7213	0.7868	0.8032
	90/10	0.6129	0.7742	0.8387	0.7742	0.6451

Amostra 70%	10-fold CV	0.7171	0.7734	0.7162	0.7586	0.7119
	70/30	0.7344	0.7343	0.75	0.7656	0.8125
	80/20	0.7907	0.6512	0.8372	0.6744	0.6279
	90/10	0.6818	0.6364	0.8182	0.7273	0.6364
Amostra 50%	10-fold CV	0.6400	0.7142	0.7471	0.7475	0.7667
	70/30	0.6957	0.7391	0.6739	0.7826	0.6739
	80/20	0.7097	0.6452	0.6129	0.7742	0.7419
	90/10	0.625	0.6875	0.6875	0.4375	0.6875
Amostra 30%	10-fold CV	0.6944	0.6722	0.6822	0.6511	0.6400
	70/30	0.6428	0.7857	0.6428	0.8214	0.7142
	80/20	0.7368	0.7368	0.6842	0.6842	0.6315
	90/10	0.8000	0.7000	0.5000	0.900	0.500

MLP						
Base	**Treinamento/Teste**	**10**	**15**	**20**	**25**	**30**
Metodologias		**Acc**	**Acc**	**Acc**	**Acc**	**Acc**
Base Original	10-fold CV	0.7262	0.7452	0.7687	0.7955	0.8183
	70/30	0.7142	0.7032	0.4505	0.7692	0.7912
	80/20	0.6885	0.8032	0.7377	0.8524	0.7869
	90/10	0.6393	0.8688	0.7704	0.8360	0.7972
Amostra 70%	10-fold CV	0.6026	0.5848	0.5783	0.6561	0.5998
	70/30	0.6250	0.6250	0.6250	0.5625	0.5781
	80/20	0.5581	0.5349	0.5349	0.6744	0.4884
	90/10	0.7727	0.4545	0.8182	0.5909	0.9091
Amostra 50%	10-fold CV	0.5629	0.7004	0.6550	0.6558	0.6146
	70/30	0.7826	0.5434	0.5434	0.7609	0.7609
	80/20	0.7742	0.8064	0.3870	0.7097	0.5806
	90/10	0.4375	0.1875	0.3871	0.6875	0.5625

Amostra 30%	10-fold CV	0.4744	0.5277	0.6044	0.6933	0.6177
	70/30	0.6071	0.6071	0.9285	0.6429	0.5000
	80/20	0.6315	0.5263	0.4737	0.5263	0.4736
	90/10	0.3000	0.8000	0.5000	0.9000	0.8000

Depois do pré-processamento:

k-NN						
Base	**Treinamento/Teste**	**1k**	**2k**	**3k**	**4k**	**5k**
Metodologias		**Acc**	**Acc**	**Acc**	**Acc**	**Acc**
Base Original	10-fold CV	0.7460	0.7690	0.7984	0.8019	0.8216
	70/30	0.7362	0.7142	0.7802	0.7912	0.8021
	80/20	0.7213	0.7049	0.7704	0.8032	0.8688
	90/10	0.8196	0.7705	0.8032	0.8032	0.8032
Amostra 70%	10-fold CV	0.7683	0.7638	0.8389	0.8342	0.8297
	70/30	0.8281	0.7187	0.7500	0.8125	0.8438
	80/20	0.8139	0.6976	0.7441	0.8372	0.7442
	90/10	0.8139	0.7674	0.7907	0.8837	0.8837
Amostra 50%	10-fold CV	0.7545	0.7550	0.7875	0.8208	0.8474
	70/30	0.7391	0.8043	0.8260	0.7608	0.8260
	80/20	0.7096	0.7096	0.8064	0.7419	0.7741
	90/10	0.8064	0.8064	0.8709	0.7741	0.8064
Amostra 30%	10-fold CV	0.7477	0.7377	0.7711	0.7377	0.7800
	70/30	0.7142	0.7857	0.8214	0.8214	0.8928
	80/20	0.6315	0.6315	0.8421	0.7368	0.8421
	90/10	0.8421	0.6842	0.7368	0.6315	0.7368

Árvore de Decisão						
Base	**Treinamento/Teste**	**Prof. 1**	**Prof. 2**	**Prof. 3**	**Prof. 4**	**Prof. 5**
Metodologias		**Acc**	**Acc**	**Acc**	**Acc**	**Acc**
Base Original	10-fold CV	0.7353	0.7390	0.7685	0.7649	0.7716
	70/30	0.7692	0.7253	0.7912	0.7912	0.6923
	80/20	0.7213	0.7213	0.7705	0.7049	0.7377
	90/10	0.6129	0.6774	0.8387	0.7419	0.6774
Amostra 70%	10-fold CV	0.7171	0.7734	0.7208	0.7539	0.7071
	70/30	0.7031	0.7656	0.7500	0.7344	0.8281
	80/20	0.7209	0.7442	0.8605	0.8372	0.6976
	90/10	0.8636	0.6818	0.7273	0.7727	0.8182
Amostra 50%	10-fold CV	0.6177	0.7075	0.7404	0.7870	0.7599
	70/30	0.7173	0.7391	0.7826	0.7608	0.5869
	80/20	0.6774	0.6774	0.7742	0.6774	0.7742
	90/10	0.5000	0.7500	0.8750	0.6875	0.9375
Amostra 30%	10-fold CV	0.6944	0.6722	0.6822	0.6733	0.6177
	70/30	0.6428	0.6428	0.8928	0.5714	0.7857
	80/20	0.6315	0.7368	0.4736	0.7894	0.7894
	90/10	0.8000	0.5000	0.6000	0.9000	0.8000

MLP normalizado						
Base	**Treinamento/Teste**	**10**	**15**	**20**	**25**	**30**
Metodologias		**Acc**	**Acc**	**Acc**	**Acc**	**Acc**
Base Original	10-fold CV	0.8385	0.8247	0.8383	0.8578	0.8415
	70/30	0.8791	0.8681	0.8681	0.8681	0.8242
	80/20	0.7869	0.8197	0.7869	0.9180	0.8033
	90/10	0.8033	0.7377	0.8361	0.9016	0.9180

Amostra 70%	10-fold CV	0.7870	0.8106	0.7865	0.8056	0.8594
	70/30	0.7813	0.7813	0.7813	0.8125	0.8593
	80/20	0.7907	0.7907	0.7907	0.7442	0.7442
	90/10	0.8636	0.7728	0.7728	0.7728	0.7273
Amostra 50%	10-fold CV	0.7942	0.7875	0.8137	0.8338	0.8075
	70/30	0.7609	0.8696	0.7391	0.7391	0.8261
	80/20	0.7419	0.7742	0.8387	0.8064	0.8065
	90/10	0.9375	0.7500	0.9375	0.8125	0.8125
Amostra 30%	10-fold CV	0.7142	0.8022	0.7711	0.8122	0.8022
	70/30	0.7143	0.8571	0.7857	0.7857	0.7143
	80/20	0.7368	0.8947	0.8421	0.7368	1.000
	90/10	0.8000	0.9000	1.000	0.7000	1.000

Naive Bayes			
Base	**Treinamento/Teste**	**Acc - Antes do processamento**	**Acc - Depois do processamento**
Metodologias			
Base Original	10-fold CV	0.7820	0.7820
	70/30	0.8021	0.8791
	80/20	0.8033	0.8032
	90/10	0.9032	0.87096
Amostra 70%	10-fold CV	0.8110	0.8103
	70/30	0.7968	0.7500
	80/20	0.6744	0.8605
	90/10	0.7727	0.7727
Amostra 50%	10-fold CV	0.8137	0.8337
	70/30	0.7391	0.7826
	80/20	0.6451	0.7742
	90/10	0.7391	0.7826

Amostra 30%	10-fold CV	0.7933	0.7822
	70/30	0.7857	0.8571
	80/20	0.7894	0.6842
	90/10	0.7000	0.7000

É possível observar que os modelos tiveram um acréscimo na performance para os parâmetros após a normalização. Principalmente os modelos como a mlp e o knn tiveram um aumento relevante nos valores de suas acurácias, como previmos ao supor estes modelos como modelos mais sensíveis à escala dos dados.

Logo, já vimos que os melhores valores foram encontrados na MLP. a seguir serão feitos comitês para testar como a utilização de vários classificadores juntos podem aprimorar os resultados. Também o naive bayes teve um aumento considerável nos valores da acurácia.

4.2 Comitês de classificadores

Neste tópico iremos detalhar a utilização dos comitês de classificadores dentro da nossa análise de dados. Para isso, utilizaremos a biblioteca *scikit-learn* do *python*.

Para cada comitê , iremos utilizar duas versões da base de dados trabalhada, a primeira versão será a versão original retirada diretamente do *kaggle,* sem nenhuma alteração estrutural, já que se trata de uma base de dados previamente tratada pelo fornecedor com relação a dados faltantes. A segunda versão é adaptada no intuito de melhorar os resultados de classificadores sensíveis a escala e também a tratamentos indevidos de “label encoding”, com dados sem comportamento ordinal, portanto os dados numéricos dessa versão foram tratados com normalização por *z-score* e os dados categóricos não binários foram tratados com “one hot encoding”.

Para cada uma das versões de dados serão criados 3 modelos de *boost* e 3 modelos de *bagging*, ambos criados levando em consideração as quantidades 10 , 15 e 20 modelos de classificadores base homogêneos. Além disso, os classificadores base serão 4, resultando em 12 modelos de *boosting* e 12 modelos de *bagging* ao todo.

Os 4 classificadores base utilizados serão: Árvore de decisão (*Decision Tree Classifier*), Naive bayes de kernel gaussiano (*GaussianNB*), KNN (K *nearest neighbors*) e a MLP (*Multi layer perceptron*).

Após os experimentos com *boosting* e *bagging* será realizado um *stacking* homogêneo seguindo o mesmo padrão do *boosting* e *bagging* com 12 modelos e 4 modelos heterogêneos escolhendo os 3 melhores classificadores nos modelos homogêneos.

4.2.1 Boosting

As considerações iniciais a respeito deste experimento de *boosting* são: inicialmente se fez um estudo de quais seriam os parâmetros mais adequados para cada um dos estimadores que seriam utilizados para a criação dos comitês. Para a MLP utilizamos a identidade de função de ativação, a função de otimização foi “sgd”(*stochastic gradient descent*), e 21 camadas escondidas, Esses parâmetros foram encontrados após a utilização de uma função de alinhamento da biblioteca

sklearn chamada “GridSearchCV”; Para o Naive Bayes utilizamos o *kernel* gaussiano; Para a árvore decisão utilizamos uma profundidade 5 que foi notada como suficiente para que o resultado fosse bom sem *overfitting*; E para o k-NN utilizamos 2 *neighbors* para os classificadores.

Outra consideração escolha feita pelo grupo foi a utilização do *boost* adaptativo, essa escolha foi feita após o estudo a respeito dos métodos de *boost* presentes na biblioteca *scikit-learn* do python, o *boost* adaptativo foi o único método que aceitava de forma eficiente todos os modelos de aprendizagem que seriam utilizados durante o experimento. Os outros métodos de *boost* encontrados foram métodos relacionados ao método do gradiente (*extreme gradient boost* e *gradient boost*), e ambos pressupõem que o classificador é a árvore de decisão. Portanto o grupo optou pela padronização do *boost* adaptativo para todos os classificadores utilizados que era possível a utilização da função de *boost*.

Outra observação importante a se fazer a respeito dessa primeira parte é que o k-NN não pode ser rodado dentro do boost adaptativo. A função “AdaBoostClassifier” pressupõe que o classificador utilizado pode ter pesos atribuídos aos modelos criados durante o “fitting”, e tanto o classificador mlp quanto o k-NN não possuem esse parâmetro de "sample weight". Para o mlp reescrevemos a classe adicionando o parâmetro de peso([**4**]StackOverFlow), entretanto para o k-NN não foi possível encontrar uma solução para a utilização de nenhum dos *boosts* do *scikit-learn*, portanto os dados do k-NN presentes na tabela de *boosting* são referentes a um único modelo que foi avaliado com cross validation 3 vezes.

Para todos os experimentos realizados neste trabalho utilizamos a função “cross_validate” utilizando a métrica de precisão macro e 10 folds para o cross validation. O resultado dessa métrica é um vetor de 10 valores de precisão correspondentes ao cross validation, e para o preenchimento da tabela fizemos a média desses 10 valores e também o desvio padrão.

Código 1- Alteração na mlp

```
class customMLPClassifer(MLPClassifier):
    def resample_with_replacement(self,
X_train,y_train,sample_weight):
        sample_weight = sample_weight
/sample_weight.sum(dtype=np.float64)
        X_train_resampled = np.zeros((len(X_train),
len(X_train[0])), dtype=np.float32)
        y_train_resampled = np.zeros((len(y_train)), dtype=np.int)
        for i in range(len(X_train)):
            draw = np.random.choice(np.arange(len(X_train)),
p=sample_weight)
            X_train_resampled[i] = X_train[draw]
            y_train_resampled[i] = y_train[draw]
        return X_train_resampled, y_train_resampled
    def fit(self, X, y, sample_weight=None):
        if sample_weight is not None:
            X, y = self.resample_with_replacement(X, y,
```

```
sample_weight)
        return self._fit(X, y, incremental=(self.warm_start and
                                            hasattr(self,
"classes_")))
```

Tabela 1 - Boosting

Boosting									
		10		15		20		média	
Base	Estimador	Média	STD	Média	STD	Média	STD	Média	STD
Base original	MLP	0.8355	0.082	0.8108	0.081	0.7987	0.057	0.82	0.073
	NB	0.8293	0.070	0.8398	0.072	0.8398	0.072	0.83	0.071
	AD	0.7917	0.069	0.7850	0.072	0.7689	0.097	0.79	0.078
	KNN	0.7999	0.092	0.8088	0.06	0.7822	0.080	0.79	0.077
	Média	0.807	0.078	0.808	0.071	0.807	0.076	X	X
Base processada	MLP	0.8568	0.061	0.8360	0.062	0.8498	0.051	0.85	0.058
	NB	0.8040	0.092	0.8040	0.092	0.8040	0.092	0.80	0.092
	AD	0.8106	0.071	0.7908	0.084	0.8038	0.071	0.80	0.075
	KNN	0.7889	0.074	0.7936	0.066	0.7895	0.071	0.80	0.07
	Média	0.807	0.0745	0.81	0.076	0.81	0.071	X	X

Com o experimento foi possível observar que alguns dos classificadores estudados são mais vulneráveis ao pré-processamento como a MLP, a árvore de decisão e o KNN. Em contrapartida é observado que o naive bayes treinou melhor modelos para os dados sem normalização, enquanto que sua precisão foi constante nos dados normalizados com um desempenho inferior aos dados originais.

É possível observar que à medida que aumentamos o número de classificadores no boost, observamos a mudança do valor entretanto, a mudança não segue um padrão generalizável, a precisão sobe para a MLP quando aumentamos o número, e o desvio padrão diminui demonstrando um modelo mais conciso, enquanto que para a árvore de decisão o aumento no número de classificadores fez com que o modelo perdesse desempenho. E para o naive bayes dos dados transformados, não houve nenhuma alteração na precisão dos classificadores. Portanto, o comportamento dos modelos de boost foram bem diversificados dependendo do classificador trabalhado.

O método aplicado retornou modelos com precisão na casa dos 80%, e essa métrica é boa para o tipo de dados que estamos trabalhando que são relacionados à saúde, mas não são excelentes dado o risco envolvido neste estudo relacionado a problemas cardíacos. Além disso, a métrica mais adequada para esse problema

talvez fosse o número de falsos positivos gerados pelo modelo, que seriam pessoas doentes que não iniciaram um tratamento por acreditarem que estão bem. Foi possível notar também que a forma de pré-processamento dos dados foi eficiente, principalmente para a convergência da MLP e para aumentar a precisão dos classificadores utilizados.

As limitações encontradas para esse método foram as restrições de funções existentes no boost adaptativo, que acabou por limitar os classificadores utilizados. Além das limitações de código, os treinos da mlp foram interrompidos antes da convergência para os dados originais e devido a isso tivemos que utilizar a “reLu” de ativação, 100 camadas escondidas e o otimizador “adam” que são os valores padrão do classificador MLP utilizado, pois variando esses valores o jupyter notebook tinha quebra de memória por atingir o limite de processamento antes da convergência. Não foi possível fazer o processo de “tuning” dos parâmetros da MLP para os dados originais devido a falta de normalização e binarização. A quantidade de dados utilizados também foi determinante para o sucesso do experimento, sendo somente 303 registros os classificadores alcançaram o resultado em tempo hábil, caso tivéssemos uma base com uma quantidade de dados muito maior os modelos da MLP provavelmente não iria convergir antes do estouro de memória no ambiente virtual utilizado para a experimentação.

4.2.2 Bagging

Para o bagging foram utilizadas as mesmas funções do boosting para os classificadores, incluindo a MLP customizada para que fosse justa em critério de comparação. Para esse comitê foi possível a utilização do KNN como classificador, portanto, todos os classificadores foram utilizados da forma proposta pela função, na metodologia proposta neste trabalho.

Para este comitê foi utilizada a função “BaggingClassifier” do scikit-learn, essa função não possui as restrições de parâmetros do peso das amostras como o boost adaptativo, e por isso foi possível a utilização do KNN e da MLP não customizada, apesar de utilizarmos a modificada para comparação justa.

Repetimos as configurações dos classificadores utilizados no boost, e o experimento permanece com os mesmo critérios do anterior com relação à metodologia de aplicação e as métricas utilizadas.

Tabela 2 - Bagging

Bagging									
		10		15		20			
Base	Estimador	Média	STD	Média	STD	Média	STD	Média	STD
Base original	MLP	0.8259	0.075	0.8416	0.066	0.8321	0.067	0.836	0.0693
	NB	0.8314	0.06	0.8184	0.06	0.8092	0.051	0.82	0.057
	AD	0.8153	0.082	0.8203	0.078	0.8249	0.078	0.813	0.0793
	KNN	0.6603	0.098	0.6663	0.099	0.6702	0.093	0.666	0.0966
	Média	0.7875	0.0788	0.785	0.0758	0.78	0.0725	X	X
Base	MLP	0.8524	0.06	0.8437	0.047	0.8473	0.055	0.85	0.054

processada	**NB**	**0.8034**	**0.082**	**0.8098**	**0.078**	**0.8169**	**0.074**	**0.813**	**0.078**
	AD	**0.8368**	**0.08**	**0.8338**	**0.085**	**0.8401**	**0.084**	**0.833**	**0.083**
	KNN	**0.8320**	**0.052**	**0.8296**	**0.052**	**0.8320**	**0.052**	**0.833**	**0.052**
	Média	**0.835**	**0.0685**	**0.825**	**0.0655**	**0.835**	**0.06625**	**X**	**X**

Para o bagging é possível notar um comportamento semelhante ao boosting com relação à aleatoriedade do comportamento dos classificadores com relação ao número de classificadores utilizados e também o comportamento do classificador com base no formato da base de dados utilizada.

Assim como no boosting a maior precisão dos dados transformados foi relacionada à MLP, além disso foi possível observar um aumento significativo no desempenho do KNN devido a transformação dos dados já que nos dados originais os valores estavam por volta dos 70% e com os dados pré processados a precisão foi para a ordem de 80%. Ao contrário do boosting, os dados originais tiveram um desempenho melhor com a MLP para o bagging em comparação ao Naive Bayes, apesar de terem valores próximos. Avaliando também o número de classificadores na função, observamos que o número 20 teve precisões mais altas para os dados transformados enquanto que os dados originais obtiveram um melhor desempenho médio com 10 classificadores. Portanto, podemos afirmar que o número de classificadores ideal varia de acordo com a base e não segue

Apesar da técnica de bagging ter sido melhor aplicada nos modelos estudados, algumas dificuldades permaneceram nessa estratégia. A MLP continua demorando para fazer o cross validation dos dados e o otimizador continua alcançando o número limite de iterações antes de convergir, o que tornaria esse classificador inviável para um conjunto de dados maior mesmo ele possuindo a maior precisão para o conjunto de dados trabalhados.

De toda forma o bagging chegou a resultados bons, assim como o boosting na casa dos 80%, com exceção do KNN para os dados originais, provavelmente devido ao fato de métodos de classificação que utilizam cálculos de distância são vulneráveis à escala dos dados.

4.2.3 Stacking homogêneo

O stacking implementado no scikit-learn é o método com abordagem mais peculiar dentre os três tipos de comitês abrangidos neste trabalho. O primeiro passo para implementação desse método é a criação de uma lista de objetos dos classificadores homogêneos que serão utilizados para a classificação. Para isso foram utilizados 3 loops “***for”*** um com 10 iterações, um com 15, e um com 20 para criar os arrays contendo os classificadores que seriam abordados. Tirando essa preparação dos classificadores base, as demais etapas do experimento seguiram os mesmo parâmetros dos comitês anteriores. Entretanto foi necessário reduzir o número de folds do cross validation para 5 para que os modelos convergissem, caso contrário dava estouro de memória por limitações computacionais.

Tabela 3 - Stacking homogêneo

Stacking Homogêneo									
		10		15		20		média	
Base	Estimador	Média	STD	Média	STD	Média	STD	Média	STD
Base original	MLP	0.8382	0.056	0.8050	0.046	0.8206	0.046	0.803	0.493
	NB	0.8092	0.064	0.8092	0.064	0.8092	0.064	0.81	0.064
	AD	0.7240	0.039	0.7468	0.04	0.7458	0.056	0.726	0.045
	KNN	0.6558	0.010	0.6558	0.010	0.6558	0.010	0.79	0.010
	Média	0.7825	0.04225	0.7825	0.04	0.7825	0.044	X	X
Base processada	MLP	0.8183	0.062	0.8253	0.05	0.8104	0.035	0.81	0.049
	NB	0.8032	0.073	0.8032	0.073	0.8032	0.073	0.8	0.073
	AD	0.7250	0.0011	0.7149	0.0084	0.7023	0.0043	0.713	0.043
	KNN	0.7963	0.010	0.7963	0.010	0.7963	0.010	0.79	0.010
	Média	0.775	0.036	0.78	0.035	0.78	0.030	X	X

O stacking homogêneo da forma como foi aplicado teve um desempenho inferior aos demais comitês, entretanto o comportamento dos melhores resultados foi semelhante ao boosting. Mais uma vez foi observado que o número ótimo de classificadores não é necessariamente o maior ou o menor, variando de classificador para classificador.

Os resultados foram semelhantes aos observados tanto no bagging quanto no boosting. A MLP permanece sendo o modelo com maior precisão para os dados processados e o Naive bayes o melhor modelo para os dados originais.

Para o stacking homogêneo o número de dificuldades foi superior aos demais, a forma de aplicação dos classificadores base nesse comitê demanda um poder computacional superior ao demandado nos comitês anteriores. Devido a isso tivemos que mudar o número de folds no cross validation para 5 caso contrário os modelos demoravam muito para convergir e tinha o risco do ambiente python ter um estouro de memória ram parando o processo.

O stacking também obteve precisões na ordem de 80% assim como os demais modelos utilizados neste trabalho, entretanto esse é inviável devido a demanda de recursos para a criação e treinamento dos modelos.

4.2.4 Stacking heterogêneo

Para o stacking homogêneo a abordagem foi diferente, devido aos desafios computacionais enfrentados no stacking homogêneo foi optado por criar arrays de

tamanho 10 para os stackings de proporção 50/50, e 15 para os stacking de proporção 33/33/33.

Os modelos base escolhidos como melhores desempenhos nos comitês anteriores foram a MLP, Naive bayes, e Árvore de decisão. Faremos portanto 4 modelos: MLP(50%)+NB(50%), AD(50%)+NB(50%), MLP(50%)+AD(50%), MLP(33%)+NB(33%)+AD(33%). e esse modelos serão treinados com o cross validation de 5 folds.

Tabela 4 - Stacking heterogêneo

Stacking Heterogêneo			
		10 10 10 (15*)	
Base	Estimador	Média	STD
Base original	MLP + NB	0.80647	0.06
	NB+AD	0.8197	0.06
	AD+MLP	0.8420	0.07
	MLP +ND+AD	0.8247	0.05
Base processada	MLP + NB	0.8481	0.05
	NB+AD	0.8441	0.1
	AD+MLP	0.7835	0.05
	MLP +ND+AD	0.8441	0.045

Foi observado um bom resultado para os *stackings* criados, boa parte deles alcançou precisões médias na ordem dos 80%. Tendo resultados equiparados à MLP desenvolvida no boost com valores de até 85% de precisão.

Além disso, é possível observar que os modelos envolvendo a MLP tiveram melhores desempenhos no geral assim como os demais comitês. e para os dados originais os modelos envolvendo Naive bayes tiveram os melhores resultados.

Mais uma vez é possível observar como a normalização e binarização podem influenciar no comportamento dos classificadores, já que os dados originais e pré-processados possuem desempenhos melhores para modelos diferentes.

Entretanto, esse comitê é inviável para uma quantidade grande de dados, o tempo e o poder computacional demandado por esse método é muito inferior quando comparado ao baixo ganho de desempenho do modelo observado.

4.3 Testes Estatísticos

Com a obtenção dos resultados fica possível realizar os testes estatísticos, determinando se as amostras vêm da mesma população (não rejeitar a hipótese nula) ou de populações diferentes (rejeitar a hipótese nula).

4.3.1 Teste de Friedman

Aplicando o teste de Friedman ao dataset original dos dados antes do processamento.

Para knn:

Teste (original)	Statistic	N	**p-value**
Friedman	8.58	5	**0.03543**

Teste (50%)	Statistic	N	**p-value**
Friedman	1.98	5	**0.57657**

Teste (30%)	Statistic	N	**p-value**
Friedman	2.94	5	**0.40097**

Para Árvore de Decisão:

Teste (original)	Statistic	n	**p-value**
Friedman	0.66	5	**0.88257**

Teste (70%)	Statistic	n	**p-value**
Friedman	1.5	5	**0.68227**

Teste (50%)	Statistic	n	**p-value**
Friedman	0.6	5	**0.89643**

Teste (30%)	Statistic	n	**p-value**
Friedman	8.76	5	**0.03266**

Depois do processamento para o ara o KNN:

Teste (original)	Statistic	n	**p-value**
Friedman	1.38	5	0.71023

Teste (70%)	Statistic	n	**p-value**
Friedman	4.38	5	**0.22325**

Teste (50%)	Statistic	n	**p-value**
Friedman	1.32	5	**0.72439**

Teste (30%)	Statistic	n	**p-value**
Friedman	1.68	5	**0.64139**

Depois do processamento para Árvore de Decisão:

Teste (original)	Statistic	n	**p-value**
Friedman	8.34	5	**0.03948**

Teste (70%)	Statistic	n	**p-value**
Friedman	0.6	5	**0.89643**

Teste (50%)	Statistic	n	**p-value**
Friedman	2.64	5	**0.45052**

Teste (30%)	Statistic	n	**p-value**
Friedman	3.24	5	**0.35608**

5. Sugestões para melhoria dos experimentos ou dificuldades encontradas

Os modelos testados tiveram um desempenho próximo ao que era esperado no início da experimentação por volta dos 80%. Entretanto, quando utilizamos a ferramenta de autoML para a base de dados inicialmente já observamos que o melhor modelo que a biblioteca encontrou para os dados utilizados foi a regressão logística e a floresta aleatória.

Desta forma, para experimentos futuros seria interessante a aplicação e criação de modelos indicados pelo autoML para testarmos os melhores parâmetros possíveis e bem como o ganho que os comitês podem atribuir ao experimento com esses modelos mais indicados.

Além disso, seria importante buscar técnicas de otimização para paralelizar as mlp de forma a melhorar o tempo de resposta do modelo dentro do ambiente de desenvolvimento python pois foi demorada a criação dos modelos multicamadas apesar do pequeno número de dados que compõem a base de dados utilizada.

6. Conclusão

O experimento foi bem sucedido ao encontrar uma acurácia na faixa dos 80% para os modelos treinados e testados durante a construção deste trabalho. Além disso, foi possível observar o quão significantes são os aumentos do desempenho dos modelos utilizando os comitês.

Durante o processo de construção do experimentos foram encontradas dificuldades de limitação tanto para linguagem python utilizada quanto para os recursos computacionais disponíveis para o grupo. Apesar disso, os resultados encontrados chegaram aos valores esperados inicialmente e se aproximaram aos valores rankeados no autoML.

Foi possível observar a importância da normalização e pré processamento dos dados trabalhados, vendo o ganho no desempenho e os resultados ultrapassando o previsto pelo autoML utilizado no tópico 3.

Para experimentos futuros é desejável a utilização de modelos mais indicados e o teste em linguagens de programação mais abrangentes para evitar possíveis problemas com os comitês.

Referências bibliográficas

[1] **Dataset Heart Disease UCI.** Disponível em: <https://www.kaggle.com/ronitf/heart-disease-uci>.Acesso: 15/08/2021
[2] **Cardiômetro.** Disponível em: <http://www.cardiometro.com.br>.Acesso: 15/08/2021

[3] **Scikit-learn: Machine Learning in Python**, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.

[4] **FIT Failed for MLP.** Disponível em: <https://stackoverflow.com/questions/60671083/gridsearchcv-fitfailedwarning-estimator-fit-failed>.Acesso: 15/08/2021

[5] **AUTOML.** Disponível em: <https://pycaret.org/>.Acesso: 15/08/2021