PEDRO HENRIQUE DOS
SANTOS DIAS LIMA

LPAA: ALGORITMOS DE MACHINE LEARNING.

POLI ESCOLA POLITÉCNICA DE PERNAMBUCO

UPE UNIVERSIDADE DE PERNAMBUCO

Dataset: Doenças cardiovasculares.

Índice.

Dataset: SAHeart

- Pesquisa realizada pelo CRFS (Coronary Risk Factor Study).
- Local: África do Sul.
- Ano: 1983.
- 462 homens brancos consultados.
- Objetivo: explorar o banco de dados para verificar se um paciente apresenta risco de contrair doenças de coração.

TRATAMENTO DE DADOS

	A
	row.names,sbp,tobacco,ldl,adiposity,famhist,typea,obesity,alcohol,age,chd
	1,160,12.00, 5.73,23.11,Present,49,25.30, 97.20,52,1
	2,144, 0.01, 4.41,28.61,Absent,55,28.87, 2.06,63,1
	3,118, 0.08, 3.48,32.28,Present,52,29.14, 3.81,46,0
	4,170, 7.50, 6.41,38.03,Present,51,31.99, 24.26,58,1
	5,134,13.60, 3.50,27.78,Present,60,25.99, 57.34,49,1
	6,132, 6.20, 6.47,36.21,Present,62,30.77, 14.14,45,0
	7,142, 4.05, 3.38,16.20,Absent,59,20.81, 2.62,38,0
	8,114, 4.08, 4.59,14.60,Present,62,23.11, 6.72,58,1
0	9,114, 0.00, 3.83,19.40,Present,49,24.86, 2.49,29,0
1	10,132, 0.00, 5.80,30.96,Present,69,30.11, 0.00,53,1
2	11,206, 6.00, 2.95,32.27,Absent,72,26.81, 56.06,60,1
3	12,134,14.10, 4.44,22.39,Present,65,23.09, 0.00,40,1
4	13,118, 0.00, 1.88,10.05,Absent,59,21.57, 0.00,17,0
5	14,132, 0.00, 1.87,17.21,Absent,49,23.63, 0.97,15,0
6	15,112, 9.65, 2.29,17.20,Present,54,23.53, 0.68,53,0

- **Sem separação de colunas**
- **Números e letras**
- **Informações irrelevantes**
- **Dataset desorganizado**

Tratamento de dados

- ✅ Separação de colunas
- ✅ Unificação de tipo de termos (N°)
- ✅ Remoção de dados irrelevantes

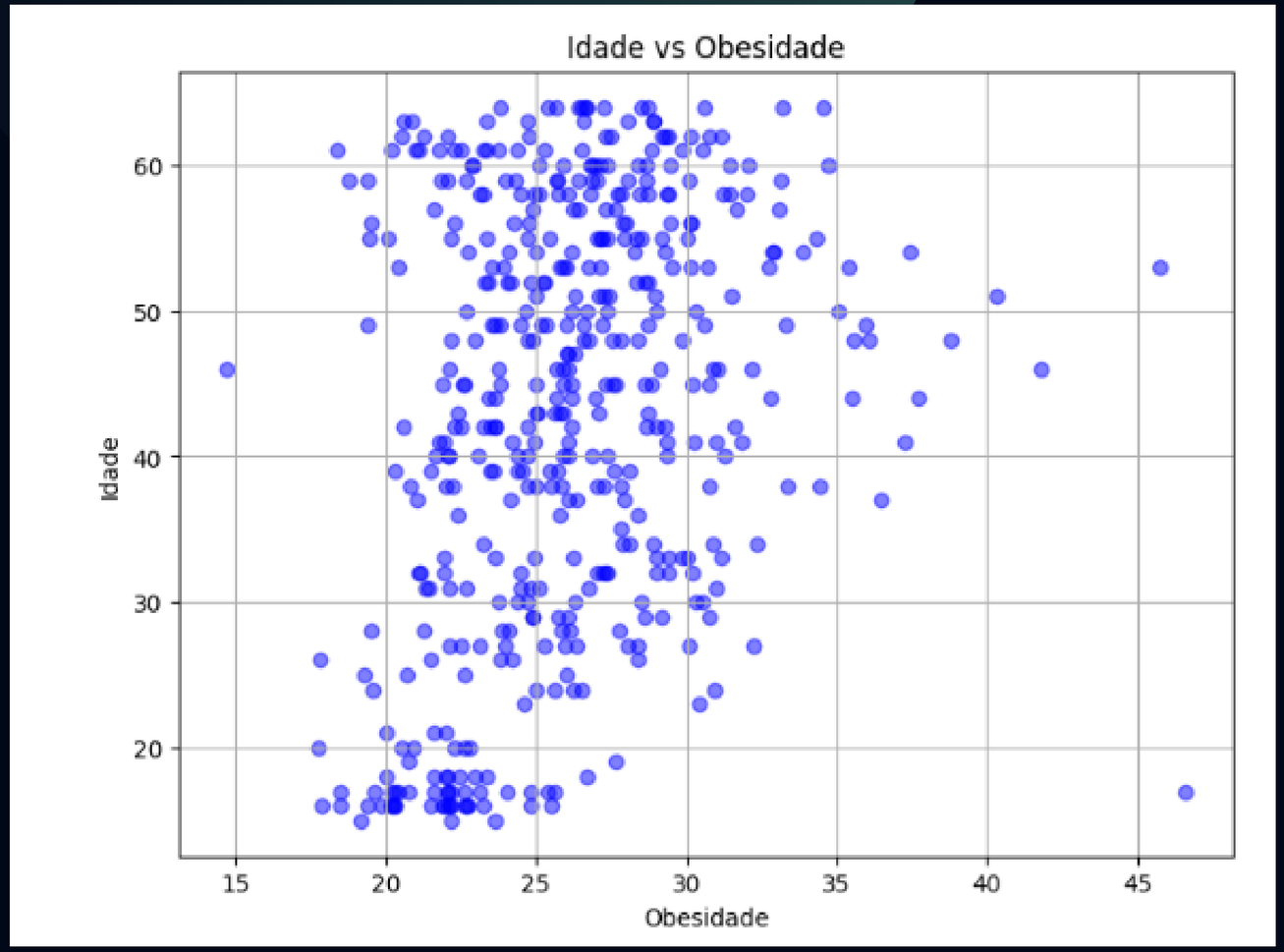
Datafarme tratado

	sbp	tobacco	ldl	adiposity	famhist	typea	obesity	alcohol	age	chd
0	160	12.00	5.73	23.11	1	49	25.30	97.20	52	1
1	144	0.01	4.41	28.61	0	55	28.87	2.06	63	1
2	118	0.08	3.48	32.28	1	52	29.14	3.81	46	0
3	170	7.50	6.41	38.03	1	51	31.99	24.26	58	1
4	134	13.60	3.50	27.78	1	60	25.99	57.34	49	1
...	...	...	...	...	...	...	...	...	...	...
457	214	0.40	5.98	31.72	0	64	28.45	0.00	58	0
458	182	4.20	4.41	32.10	0	52	28.61	18.72	52	1
459	108	3.00	1.59	15.23	0	40	20.09	26.64	55	0
460	118	5.40	11.61	30.79	0	64	27.35	23.97	40	0
461	132	0.00	4.82	33.41	1	62	14.70	0.00	46	1

Desta maneira, conseguimos dar início ao estudo do conjunto de dados.

Análise de dados

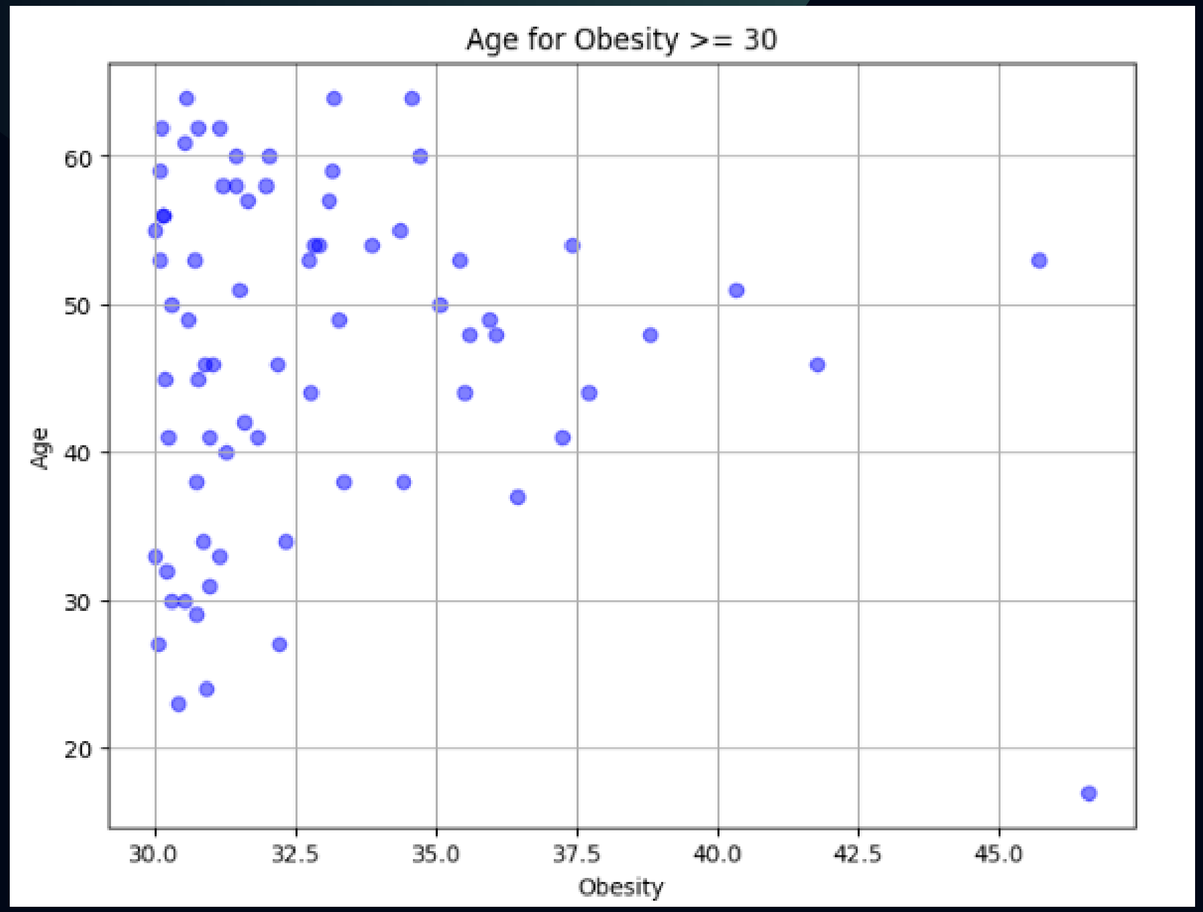
Gráfico de dispersão



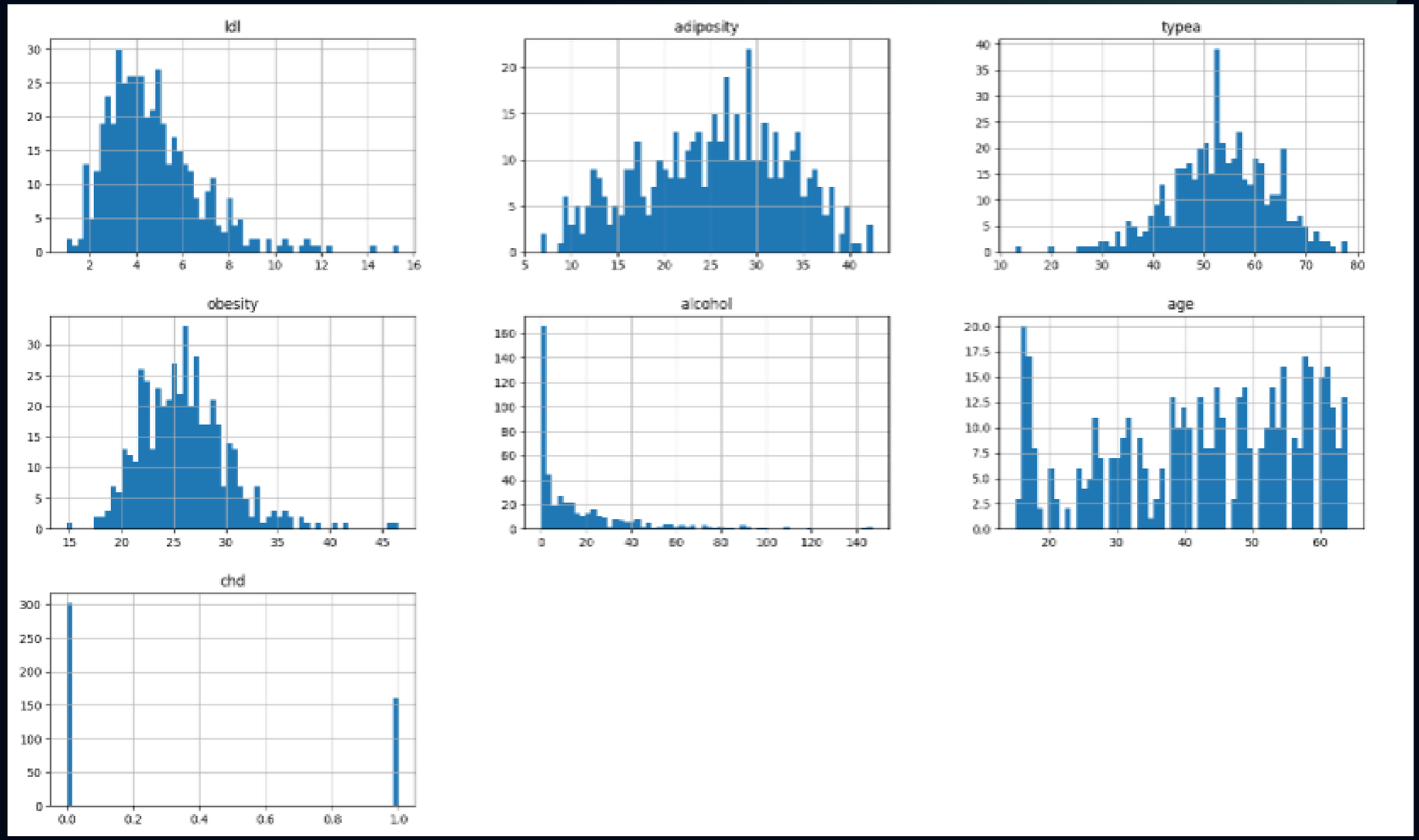
Análise de dados

Idade de pessoas que estão com IMC acima de 30 (Obesos)

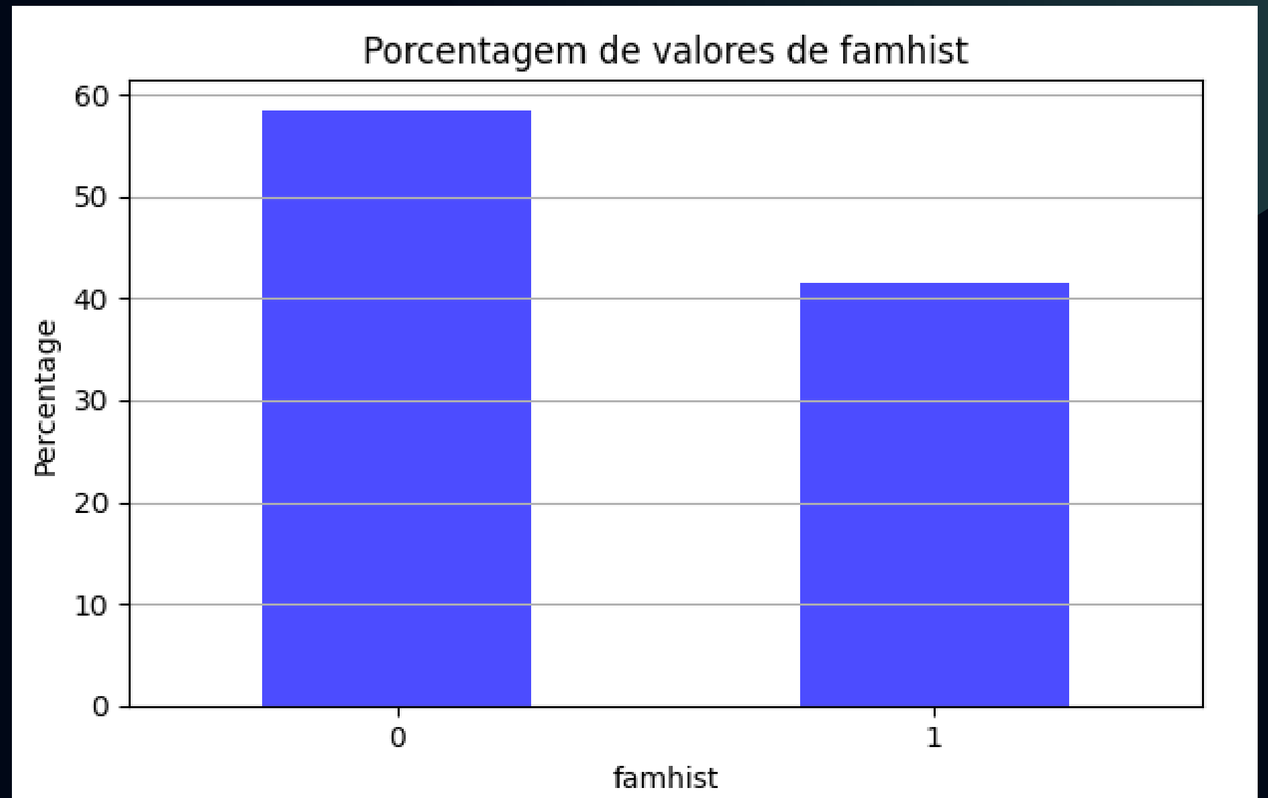
Gráfico de dispersão



Histograma com frequência de cada coluna

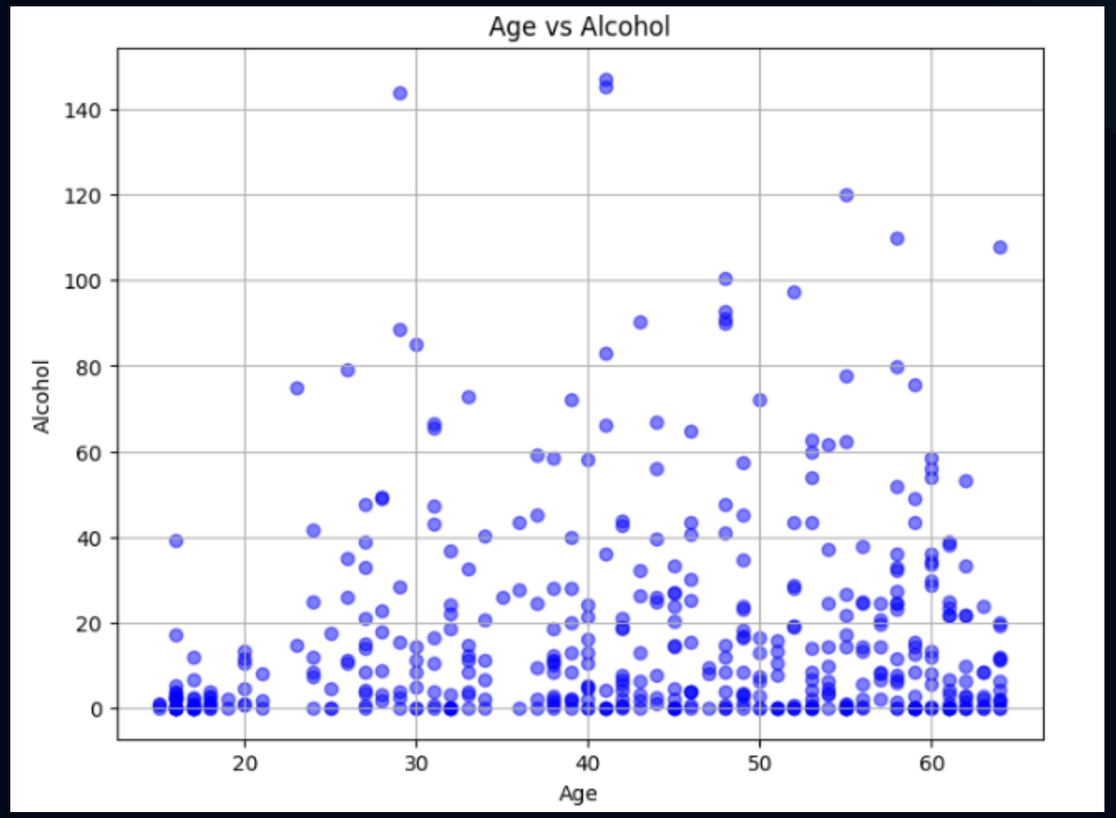


Porcentagem de presença de doenças cardíacas no histórico familiar




```
Porcentagem de '1' na coluna 'famhist': 41.55844155844156
Porcentagem de '0' na coluna 'famhist': 58.44155844155844
```

Idade de pessoas que ingerem alcool



Pessoas menores de idade que ingerem álcool

13	0.97
48	38.98
64	2.06
70	0.60
110	2.49
156	1.03
196	0.51
260	0.26
268	17.14
285	2.42
288	2.78
290	3.50
291	5.19
330	11.83
432	2.06
434	3.87
435	2.06
436	6.51
437	2.49

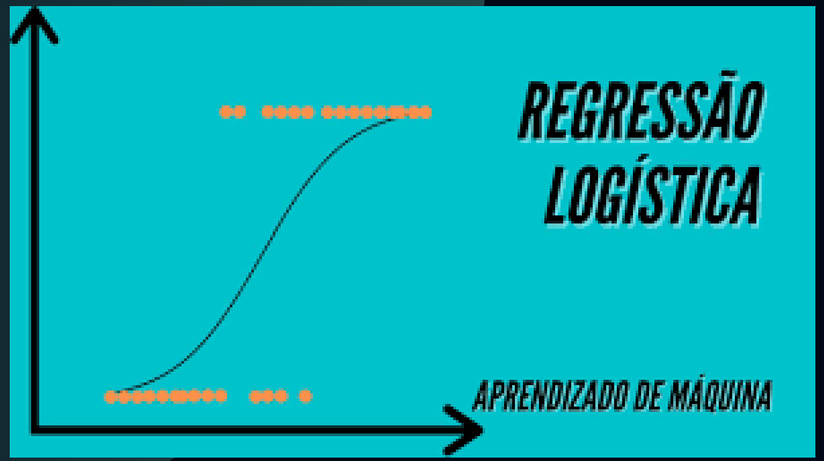
Treinamentos e testes

```
#Os codigos anteriores, eram referentes a tratamento de dados conforme a primeira unidade.
#Agora partiremos para segunda unidade, onde exploramos ferramentas de Machine Learning


#Essa parte do código fará uma divisão dos dados em conjuntos de treinamento e teste usando a função train_test_split do scikit-learn (sklearn)

from sklearn.model_selection import train_test_split #importando a funcao
#agora dividindo os dados em conjunto de treinamento e teste:

X_train,X_test,y_train,y_test = train_test_split(X,y,train_size=0.8)
#x é o conjunto de features e y o conjunto de rotulos.
#o train_size especifica a porcentagem de dados que serao usados para treinamento (80%), restando 20% para teste.
```

```
#no codigo abaixo, aplicaremos duas técnicas de pré-processamento de dados aos conjuntos de treinamento e teste.
#utilizaremos as classes StandardScaler e MinMaxScaler do scikit-learn (sklearn).

from sklearn.preprocessing import StandardScaler, MinMaxScaler #importando as classes

scale_columns = [col for col in X.columns if col!='famhist'] #definindo as colunas escaladas, excluindo "famhist"
scaler = StandardScaler() #objeto standscaler
min_max_scaler = MinMaxScaler() #objeto minmaxscaler

X_train[scale_columns] = scaler.fit_transform(X_train[scale_columns]) #padronizando colunas de train com standscaler
X_test[scale_columns] = scaler.transform(X_test[scale_columns]) #padronizando colunas de test com standscaler

#agora aplica a escala min-max (normalização) às mesmas colunas, tanto nos conjuntos de train quanto nos test, usando MinMaxScaler
X_train[scale_columns] = min_max_scaler.fit_transform(X_train[scale_columns])
X_test[scale_columns] = min_max_scaler.transform(X_test[scale_columns])
```

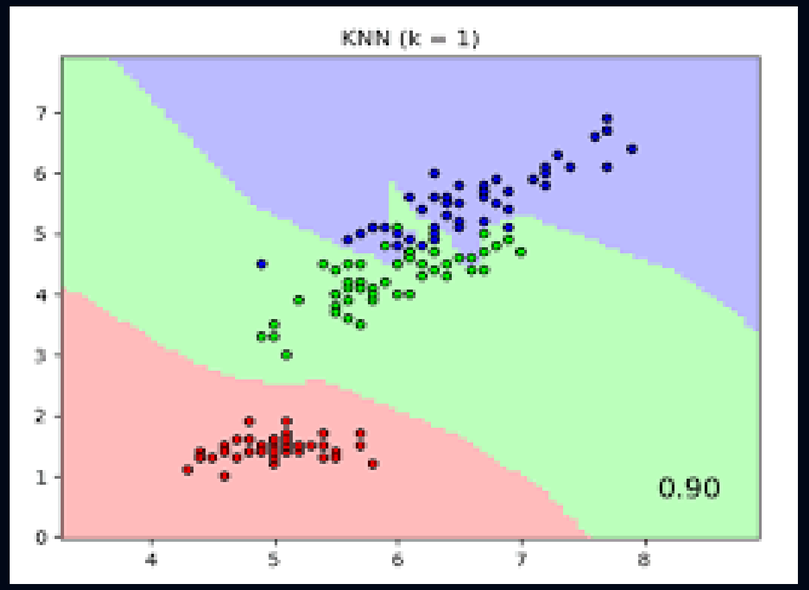
80% dos dados destinados a treinamento e 20% destinados a teste.

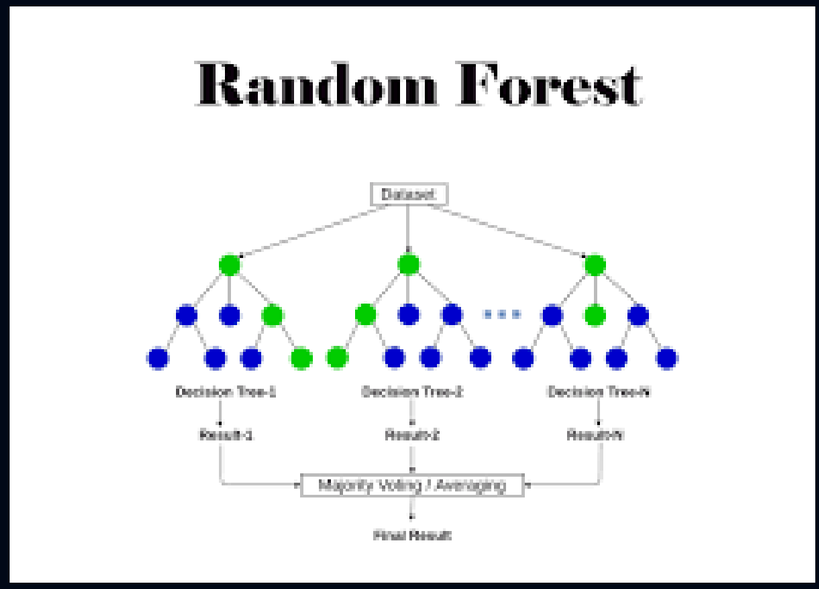
Tipos de Algoritmos



- Regressão Logística.
- Random Forest
- KNN

Nesse caso, o objetivo seria prever com base nos dados informados a probabilidade de uma pessoa ter doença cardiovascular.





Algoritmos

Regressão Logística

- Iteração máxima de 300

Resultados obtidos

```
accuracy score :  0.7419354838709677
precision_score :  0.5
recall score :  0.5833333333333334
f1_score :  0.5384615384615384
roc_auc_score :  0.6902173913043479
```

Algoritmos

Random Forest

Resultados obtidos

```
accuracy score :  0.6344086021505376
precision_score :  0.3928571428571428
recall score :  0.3928571428571428
f1_score :  0.3928571428571428
roc_auc_score :  0.5656593406593408
```

Algoritmos

K-N-N

Resultados obtidos

```
accuracy score :  0.6666666666666666
precision_score :  0.3928571428571428
recall score :  0.44
f1_score :  0.4150943396226415
roc_auc_score :  0.5950000000000001
```

Comparação de resultados:

Regressão Logística

```
accuracy score :  0.7419354838709677
precision_score :  0.5
recall score :  0.5833333333333334
f1_score :  0.5384615384615384
roc_auc_score :  0.6902173913043479
```

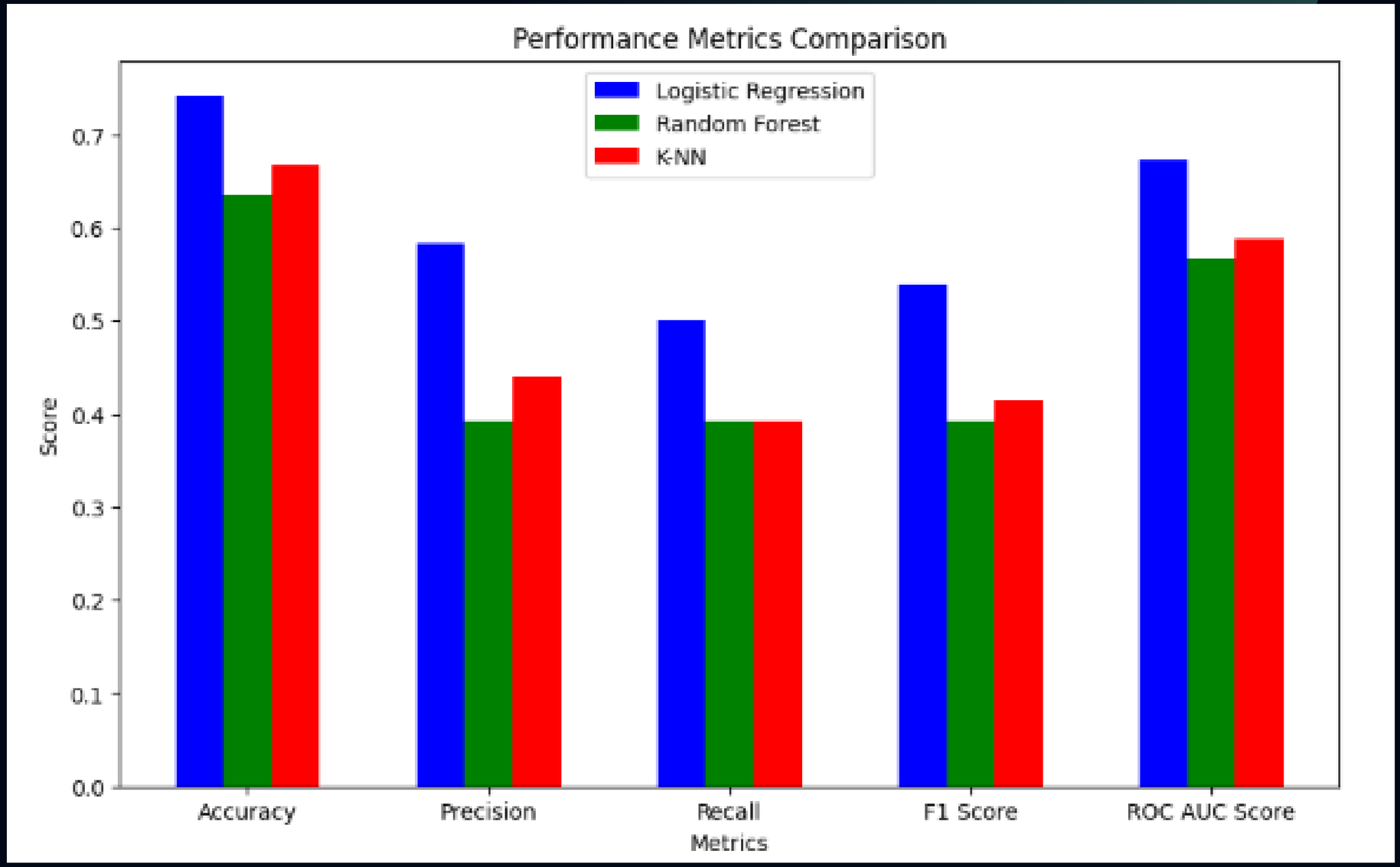
Random Forest

```
accuracy score :  0.6344086021505376
precision_score :  0.3928571428571428
recall score :  0.3928571428571428
f1_score :  0.3928571428571428
roc_auc_score :  0.5656593406593408
```

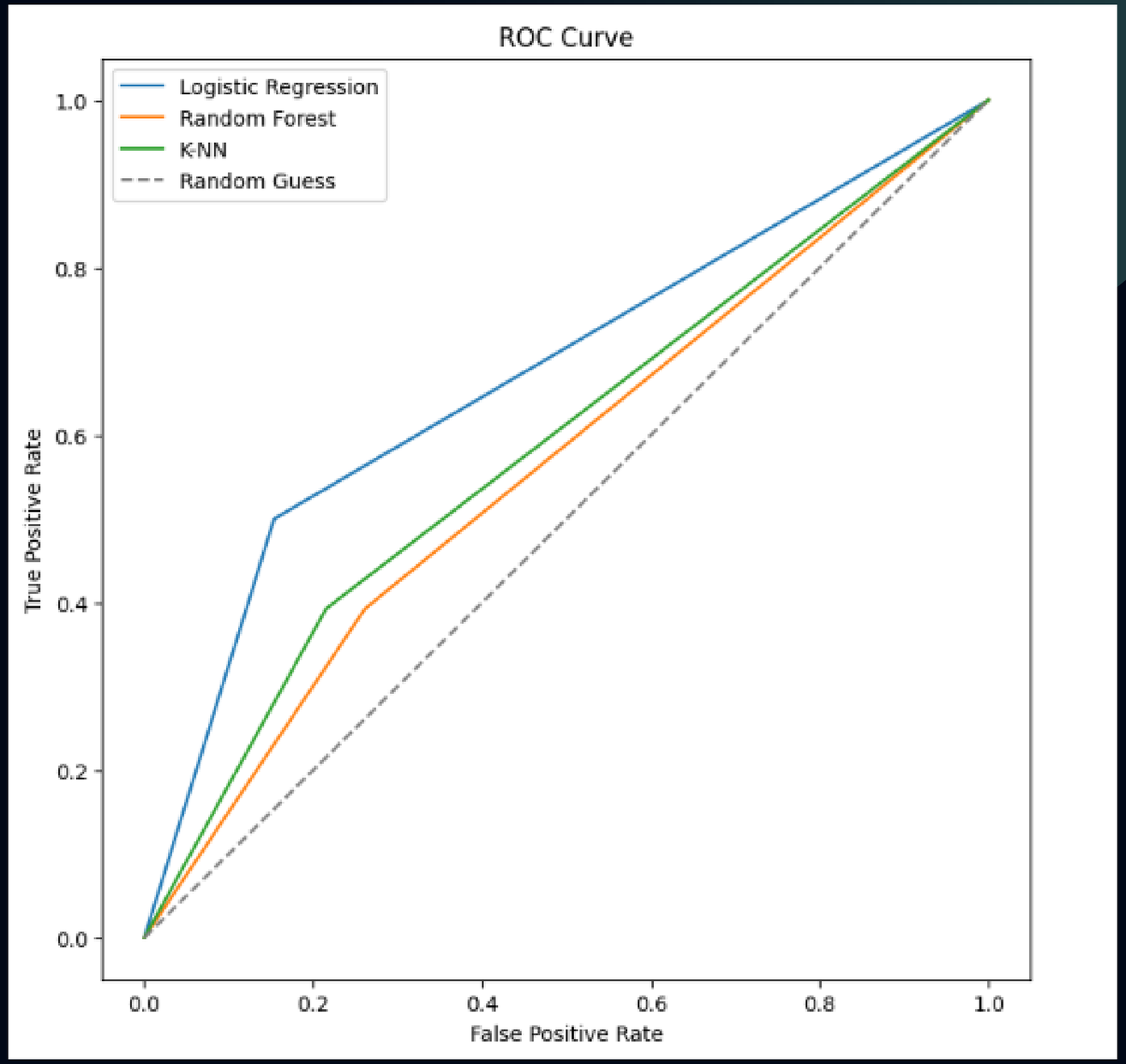
K-N-N

```
accuracy score :  0.6666666666666666
precision_score :  0.3928571428571428
recall score :  0.44
f1_score :  0.4150943396226415
roc_auc_score :  0.5950000000000001
```

Gráfico de comparação de resultados:



Plotagem da Curva ROC



Algoritmo escolhido