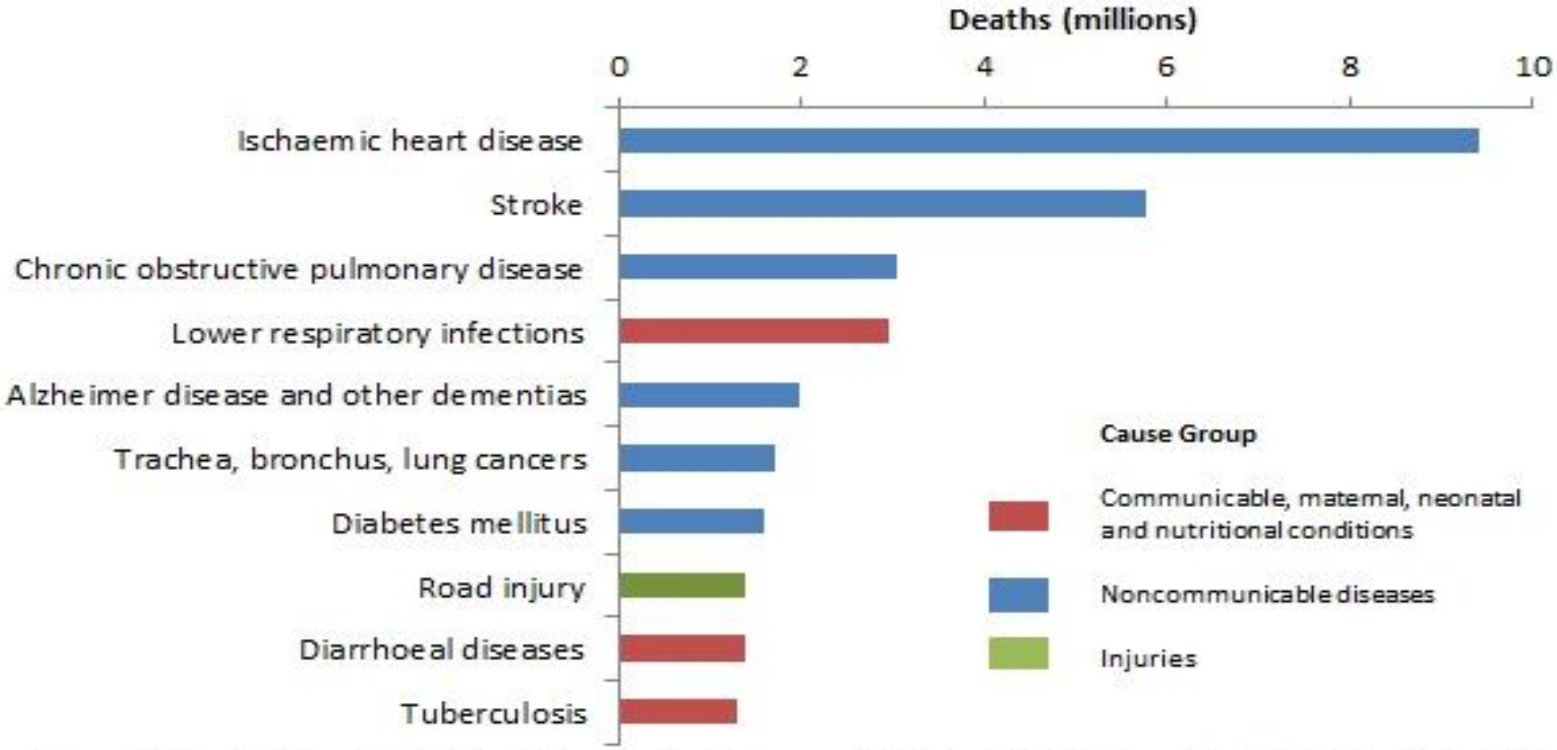
Preditor de risco de problema cardiovascular

Anna Beatriz
Stenio Ellison

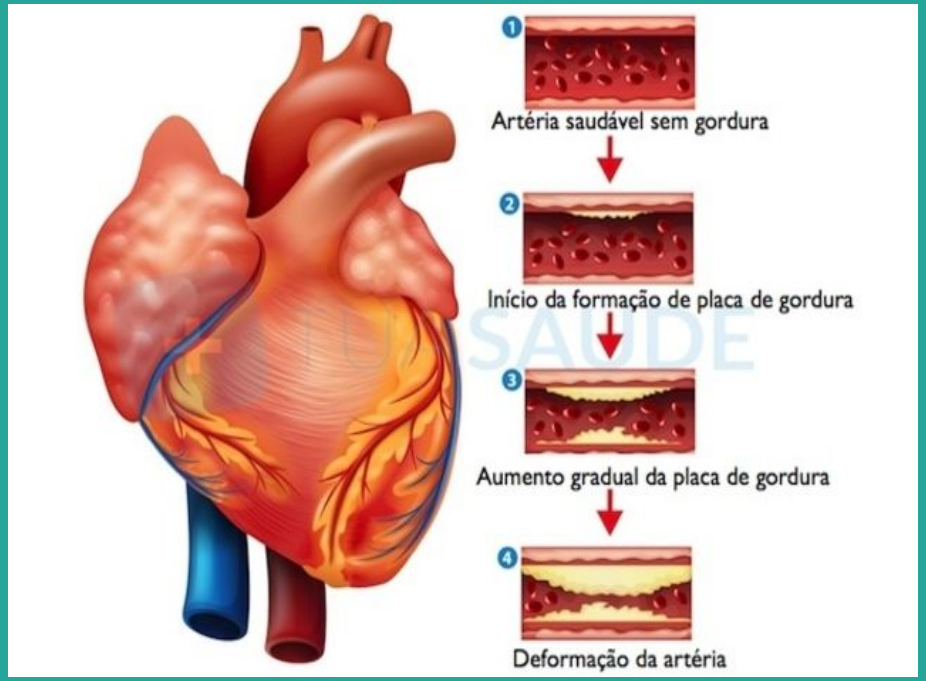
Motivação

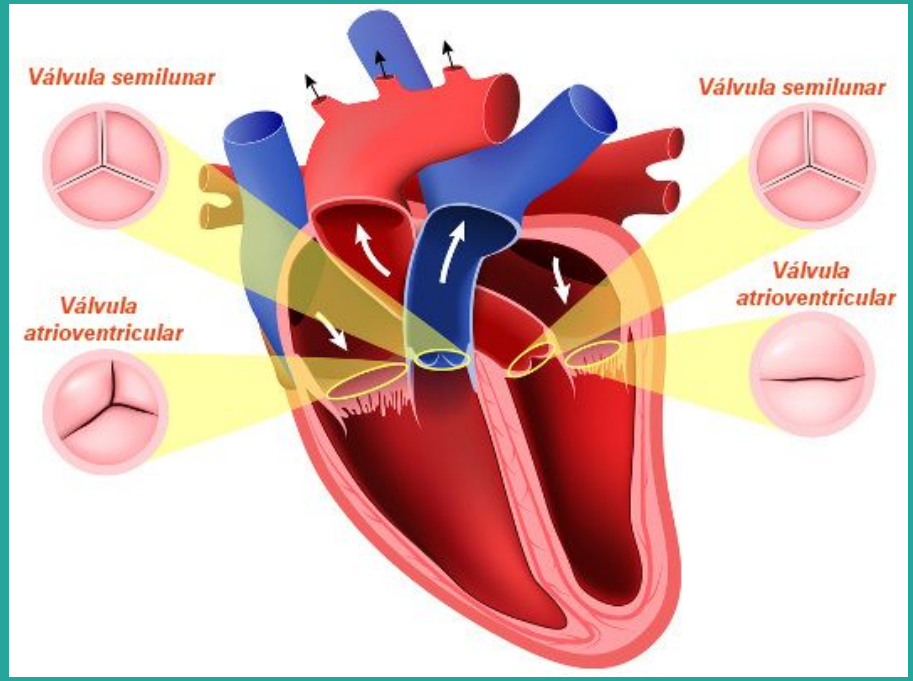
- Doenças cardiovasculares é a principal causa de morte do mundo
- Existe uma enorme quantidades de dados relacionados a tais doenças
- Existe uma demanda de redes hospitalares em relação a análise de tais dados

Top 10 global causes of deaths, 2016



Source: Global Health Estimates 2016: Deaths by Cause, Age, Sex, by Country and by Region, 2000-2016. Geneva, World Health Organization; 2018.


1
Artéria saudável sem gordura
2
Início da formação de placa de gordura
3
Aumento gradual da placa de gordura
4
Deformação da artéria


Válvula semilunar
Válvula semilunar
Válvula atrioventricular
Válvula atrioventricular

Base de dados - Heart Disease UCI

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

303 rows × 14 columns

Informações dos atributos

Age: Idade em anos
sex: 1 masculino; 0 feminino
cp: Tipo de dor no peito(Angina típica, angina atípica, não-angina ou angina assintomática)
trestbps: Pressão arterial (em mm Hg na admissão no hospital)
Chol: Colesterol
fbs: Açúcar no sangue em jejum
restecg: Resultados eletrocardiográficos em repouso
thalach: Máx. frequência cardíaca atingida durante o teste de estresse com tálio
exang: Angina induzida pelo exercício (1 sim; 0 não)
oldpeak: Depressão do segmento ST induzida pelo exercício
slope: Declive (Slope) do pico do segmento ST do exercício (0 = subida, 1 = plana ou 2 = descida)
ca: Número de vasos principais (0-3) coloridos por flourosopy 4 = NA
thal: Resultado do teste de estresse do thalium 3 = normal; 6 = defeito fixo; 7 = defeito reversível 0 = NA
target: Status da doença cardíaca 1 ou 0 (0 = doença cardíaca 1 = assintomática)

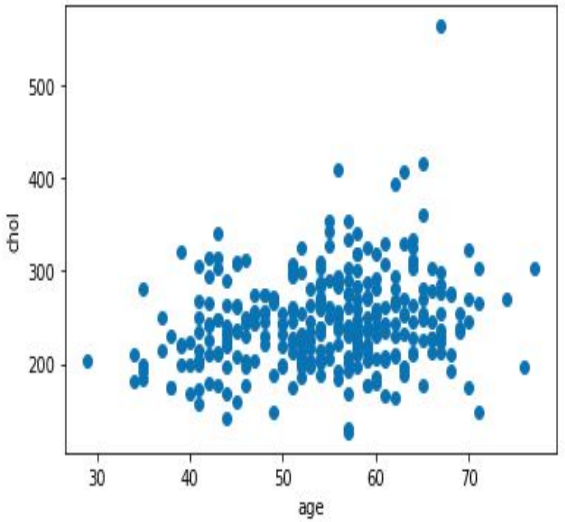
Pré-processamento

```
df.mean() #cálculo da media
```

```
age          54.366337
sex           0.683168
cp            0.966997
trestbps    131.623762
chol        246.264026
fbs           0.148515
restecg       0.528053
thalach     149.646865
exang         0.326733
oldpeak       1.039604
slope         1.399340
ca            0.729373
thal          2.313531
target        0.544554
dtype: float64
```

```
[ ] plt.scatter(df.age, df.chol) #gráfico de correlação idade x colesterol
    plt.xlabel('age')
    plt.ylabel('chol')
```

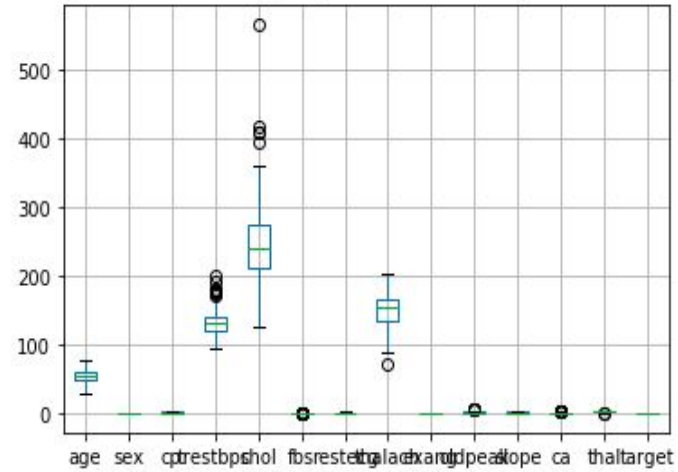
```
Text(0, 0.5, 'chol')
```



Pré-processamento

```
[ ] df.iloc[:,12] = df.iloc[:,12:]
    %matplotlib inline

    df.boxplot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f24a16eb390>
```




```
print('VF:', df.isnull().sum()) #verificando campos vazios e dados duplicados
print('VD:', df.duplicated())
```

```
VF: age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
VD: 0      False
1      False
2      False
3      False
4      False
       ...
298    False
299    False
300    False
301    False
302    False
Length: 303, dtype: bool
```

Pré-processamento

```
[ ] corr = df.corr() #gerando matriz de correlação
    corr.style.background_gradient(cmap='magma')
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
age	1	-0.0984466	-0.068653	0.279351	0.213678	0.121308	-0.116211	-0.398522	0.0968008	0.210013	-0.168814	0.276326	0.0680014	-0.225439
sex	-0.0984466	1	-0.0493529	-0.0567688	-0.197912	0.0450318	-0.0581963	-0.0440199	0.141664	0.0960929	-0.0307106	0.118261	0.210041	-0.280937
cp	-0.068653	-0.0493529	1	0.0476078	-0.0769044	0.094444	0.0444206	0.295762	-0.39428	-0.14923	0.119717	-0.181053	-0.161736	0.433798
trestbps	0.279351	-0.0567688	0.0476078	1	0.123174	0.177531	-0.114103	-0.0466977	0.0676161	0.193216	-0.121475	0.101389	0.0622099	-0.144931
chol	0.213678	-0.197912	-0.0769044	0.123174	1	0.0132936	-0.15104	-0.00993984	0.0670228	0.0539519	-0.00403777	0.0705109	0.098803	-0.0852391
fbs	0.121308	0.0450318	0.094444	0.177531	0.0132936	1	-0.0841891	-0.00856711	0.0256651	0.00574722	-0.0598942	0.137979	-0.0320193	-0.0280458
restecg	-0.116211	-0.0581963	0.0444206	-0.114103	-0.15104	-0.0841891	1	0.0441234	-0.0707329	-0.0587702	0.0930448	-0.0720424	-0.0119814	0.13723
thalach	-0.398522	-0.0440199	0.295762	-0.0466977	-0.00993984	-0.00856711	0.0441234	1	-0.378812	-0.344187	0.386784	-0.213177	-0.0964391	0.421741
exang	0.0968008	0.141664	-0.39428	0.0676161	0.0670228	0.0256651	-0.0707329	-0.378812	1	0.288223	-0.257748	0.115739	0.206754	-0.436757
oldpeak	0.210013	0.0960929	-0.14923	0.193216	0.0539519	0.00574722	-0.0587702	-0.344187	0.288223	1	-0.577537	0.222682	0.210244	-0.430696
slope	-0.168814	-0.0307106	0.119717	-0.121475	-0.00403777	-0.0598942	0.0930448	0.386784	-0.257748	-0.577537	1	-0.0801552	-0.104764	0.345877
ca	0.276326	0.118261	-0.181053	0.101389	0.0705109	0.137979	-0.0720424	-0.213177	0.115739	0.222682	-0.0801552	1	0.151832	-0.391724
thal	0.0680014	0.210041	-0.161736	0.0622099	0.098803	-0.0320193	-0.0119814	-0.0964391	0.206754	0.210244	-0.104764	0.151832	1	-0.344029
target	-0.225439	-0.280937	0.433798	-0.144931	-0.0852391	-0.0280458	0.13723	0.421741	-0.436757	-0.430696	0.345877	-0.391724	-0.344029	1

Treino e Teste

```
[ ] from sklearn.model_selection import train_test_split

    X=df[['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal']] # Features
    y=df['target']  # Labels

    # Split dataset into training set and test set
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3) # 70% training and 30% test
```

70% 30%

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604	1.399340	0.729373	2.313531	0.544554
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226	1.022606	0.612277	0.498835
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000	2.000000	0.000000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000	0.000000	2.000000	1.000000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000	3.000000	1.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3.000000	1.000000

Dataset describe

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
count	212.000000	212.000000	212.000000	212.000000	212.000000	212.000000	212.000000	212.000000	212.000000	212.000000	212.000000	212.000000	212.000000
mean	54.311321	0.693396	0.929245	131.787736	244.226415	0.136792	0.523585	149.702830	0.306604	0.984906	1.419811	0.683962	2.297170
std	9.083982	0.462175	1.025594	16.635691	46.268490	0.344441	0.528263	22.432868	0.462175	1.135117	0.606586	1.006673	0.616819
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	47.000000	0.000000	0.000000	120.000000	211.750000	0.000000	0.000000	135.500000	0.000000	0.000000	1.000000	0.000000	2.000000
50%	56.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.600000	1.000000	0.000000	2.000000
75%	61.000000	1.000000	2.000000	140.000000	271.000000	0.000000	1.000000	165.000000	1.000000	1.600000	2.000000	1.000000	3.000000
max	77.000000	1.000000	3.000000	200.000000	409.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3.000000

Train describe

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
count	212.000000	212.000000	212.000000	212.000000	212.000000	212.000000	212.000000	212.000000	212.000000	212.000000	212.000000	212.000000	212.000000
mean	54.311321	0.693396	0.929245	131.787736	244.226415	0.136792	0.523585	149.702830	0.306604	0.984906	1.419811	0.683962	2.297170
std	9.083982	0.462175	1.025594	16.635691	46.268490	0.344441	0.528263	22.432868	0.462175	1.135117	0.606586	1.006673	0.616819
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	47.000000	0.000000	0.000000	120.000000	211.750000	0.000000	0.000000	135.500000	0.000000	0.000000	1.000000	0.000000	2.000000
50%	56.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.600000	1.000000	0.000000	2.000000
75%	61.000000	1.000000	2.000000	140.000000	271.000000	0.000000	1.000000	165.000000	1.000000	1.600000	2.000000	1.000000	3.000000
max	77.000000	1.000000	3.000000	200.000000	409.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3.000000

Test describe

Os conjuntos de dados são representativos no treino e teste, pois não houve uma discrepância grande referente ao describe da base de dados original.

Random Forest

Random Forest

Um dos mais populares e potentes algoritmos supervisionado do machine learning capaz de executar tarefas de regressão ou classificação, sugere criar uma “floresta” com o número das árvores de decisão , quanto mais árvores de decisão na “floresta” mais robusta se torna a predição e acurácia.

A motivação para usar este algoritmo está em seu desempenho em predizer de forma confiável o resultado e o por ser capaz de identificar doenças analisando os testes do paciente

Iniciamos com um número de 42 estimadores e obtvemos melhor resultado com 150 estimadores, esses estimadores são o conjunto de vários modelos de árvores de decisão e 50 conjugados onde o bootstrap trata os dados, como insere, remove ou armazena para o algoritmo ter uma predição .

Random Forest

```
[ ]  #Import Random Forest Model
     from sklearn.ensemble import RandomForestClassifier
     clfRF=RandomForestClassifier(n_estimators=150, random_state = 50, bootstrap='True')
     clfRF.fit(X_train,y_train)
     y_pred=clfRF.predict(X_test)
     #Import scikit-learn metrics module for accuracy calculation
     from sklearn import metrics
     print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
     print("Precision:",metrics.precision_score(y_test, y_pred))
     print("Recall:",metrics.recall_score(y_test, y_pred))
     from sklearn.metrics import classification_report, confusion_matrix
     from sklearn.metrics import multilabel_confusion_matrix
     cm=confusion_matrix(y_test, y_pred)#A matriz de confusão com as classes concatenadas
     print(cm)
```

Random Forest

```
Accuracy: 0.8131868131868132
Precision: 0.8043478260869565
Recall: 0.8222222222222222
[[37  9]
 [ 8 37]]
```

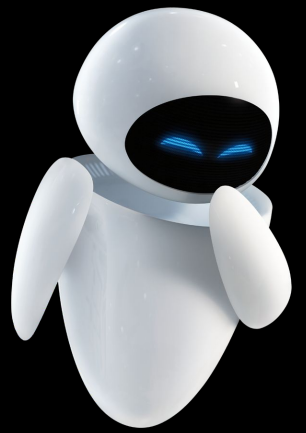
SVM

O SVM (do inglês “Support Vector Machine”) é uma técnica supervisionada de aprendizado de máquina amplamente usada em problemas de reconhecimento e classificação de padrões,possui alta precisão, boas garantias teóricas sobre o ajuste excessivo e, com um kernel apropriado, elas podem funcionar bem, mesmo que os dados não sejam linearmente separáveis no espaço de recursos base.

São ótimos algoritmos a ser implementados em problemas com 2 classes, o kernel linear utilizado ofereceu uma melhor formação de hiperplano para separação dessas duas classes (tendo resultado comparado com outros kernels).

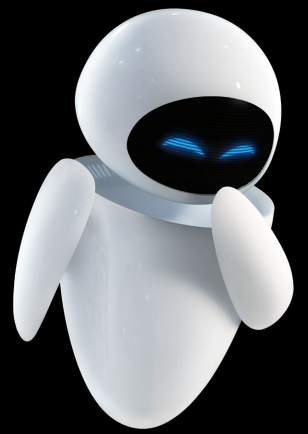
SVM

```
#Import svm model
from sklearn import svm
clfSVM = svm.SVC(kernel='linear') # Linear Kernel
clfSVM.fit(X_train, y_train)
y_pred = clfSVM.predict(X_test)

print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
print("Precision:",metrics.precision_score(y_test, y_pred))
print("Recall:",metrics.recall_score(y_test, y_pred))
cm=confusion_matrix(y_test, y_pred)#A matriz de confusão com as classes concatenadas
print(cm)
```



SVM

```
Accuracy: 0.8131868131868132
Precision: 0.8275862068965517
Recall: 0.8727272727272727
[[26 10]
 [ 7 48]]
```



CatBoost

CatBoost

O CatBoost também é um algoritmo machine learning popular e potente, ele utiliza árvore de decisão e vem sendo mais utilizado que o Random Forest por ter um método de otimização conhecido como gradiente boosting capaz de tornar fracos aprendedores em fortes aprendedores nas árvores de decisões.

Esta biblioteca é baseada na biblioteca de aumento de gradiente. O aumento de gradiente é um poderoso algoritmo de aprendizado de máquina que é amplamente aplicado a vários tipos de desafios de negócios, como detecção de fraudes, itens de recomendação, previsão e apresenta bom desempenho.Também pode retornar resultados muito bons com relativamente menos dados, ao contrário dos modelos de DL que precisam aprender com uma quantidade enorme de dados. Ele produz resultados de ponta, sem treinamento extensivo de dados, normalmente exigido por outros métodos de aprendizado de máquina

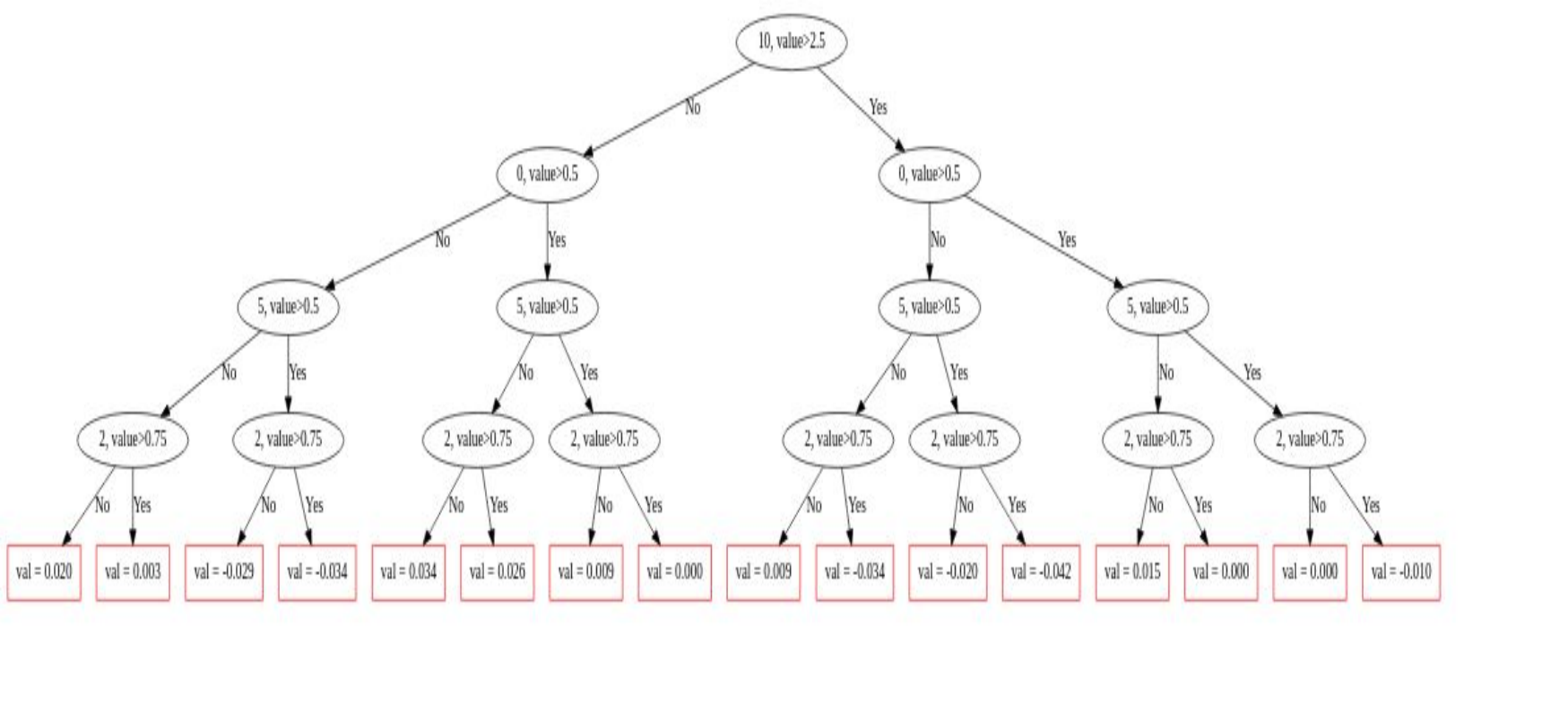
CatBoost

```
[ ] from catboost import CatBoostClassifier
    # Initialize data
    X = df[set(df.columns)-set(['target'])].values
    y = df.target.values

    from sklearn.model_selection import train_test_split
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
    X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size = 0.2, random_state = 0)

    #Importando o classificador e fazendo o modelo
    from catboost import CatBoostClassifier
    model=CatBoostClassifier(iterations=300, depth=4, learning_rate=0.01, loss_function='CrossEntropy', task_type='GPU')
    model.fit(X_train, y_train,eval_set=(X_val, y_val),plot=True)
```

CatBoost

```
Precision: 0.8823529411764706
Recall: 0.8823529411764706
Acurácia: 86.88524590163934
[[23  4]
 [ 4 30]]
```

CatBoost


10, value>2.5
No
Yes
0, value>0.5
0, value>0.5
No
Yes
No
Yes
5, value>0.5
5, value>0.5
5, value>0.5
5, value>0.5
No
Yes
No
Yes
No
Yes
No
Yes
2, value>0.75
2, value>0.75
2, value>0.75
2, value>0.75
2, value>0.75
2, value>0.75
2, value>0.75
2, value>0.75
No
Yes
No
Yes
No
Yes
No
Yes
No
Yes
No
Yes
No
Yes
No
Yes
val = 0.020
val = 0.003
val = -0.029
val = -0.034
val = 0.034
val = 0.026
val = 0.009
val = 0.000
val = 0.009
val = -0.034
val = -0.020
val = -0.042
val = 0.015
val = 0.000
val = 0.000
val = -0.010

```
[ ] model.get_feature_importance(data=None,
                                 prettified=False,
                                 thread_count=-1,
                                 verbose=False)
```

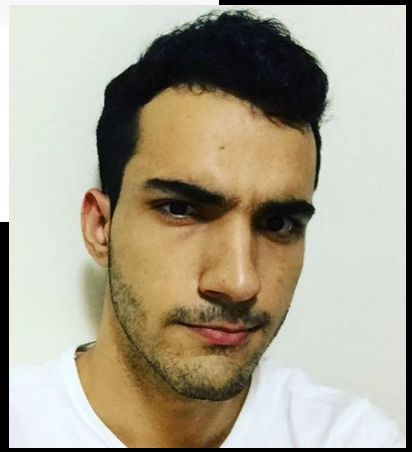
```
array([ 4.02324941,  5.01077386,  2.48789337,  2.61655172, 26.68661662,
        1.9282344 ,  8.06580386,  0.97322999,  5.66139618,  7.23583383,
        7.88912868, 10.88999709, 16.53129099])
```

O CatBoost possui a função acima que permite saber o atributo mais importante entre as outros, nesse array pode-se visualizar que o colesterol tem importância maior, o que faz sentido tratando de problemas cardiovascular. O cálculo feito pela função compara a variação do valor dos atributos com a média de predição, assim classificando o atributo como o mais importante.

Teste com novos dados

```
#Teste com novo conjunto dados
flavio = [[21, 1, 2, 120, 123, 0, 1, 180, 0, 0, 1, 1, 2]]
print(clfSVM.predict(flavio)) #SVM
print(clfRF.predict(flavio)) #randomforest
print(model.predict(flavio)) #catboost
```

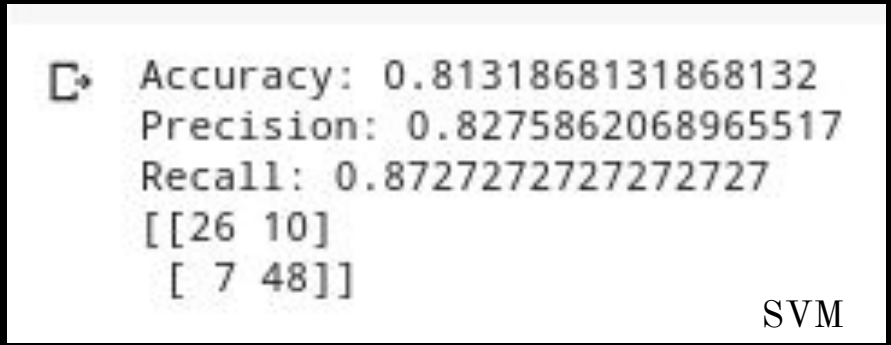
```
[1]
[1]
[1]
```



Aqui podemos comparar os 3 algoritmo (onde todos obtiveram acurácia e precisão entre 80-90%) com um novo conjunto de dados de um voluntário em que houve a mesma classificação para os 3

Resultados

Como mencionado obtivemos resultados em intervalos semelhantes para ambos os algoritmos com treino e teste a 70% e 30% e supermercad e uma diferença parcial apenas no catboost para treino e teste 80% e 20%.



SVM

Random Forest X CatBoost

```
Precision: 0.8823529411764706
Recall: 0.8823529411764706
Acurácia: 86.88524590163934
[[23  4]
 [ 4 30]]
```

```
Accuracy: 0.8131868131868132
Precision: 0.8043478260869565
Recall: 0.8222222222222222
[[37  9]
 [ 8 37]]
```

O Random Forest assim como CatBoost utiliza de árvore decisão em seus métodos, porém o CatBoost obteve performance melhor pois possui um método de otimização de Gradiente descendente.

Referências

https://scikit-learn.org/stable/supervised_learning.html#supervised-learning

https://www.kaggle.com/ronitf/heart-disease-uci