

UNIT

SISTEMA DA INFORMAÇÃO

TACYANNE BERNADETE LIMA PIMENTEL

RELATÓRIO DA MEDIDA DE EFICIÊNCIA

UNIT/SE

ARACAJU

2019

TACYANNE BERNADETE LIMA PIMENTEL

RELATÓRIO DA MEDIDA DE EFICIÊNCIA

UNIT/SE

Relatorio apresentado ao Prof.Msc. Fabio Gomes Rocha da Universidade Tiranders como forma de avaliação desenvolvidas durante o 2° semestre.

ARACAJU

2019

SUMÁRIO

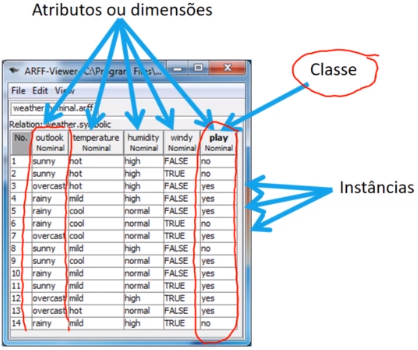
1. INTRODUÇÃO
2. ATIVIDADES DESENVOLVIDAS
3. CONCLUSÃO
4. REFERÊNCIAS
5. INTRODUÇÃO

Machine Learning consiste, no aprendizado de máquina, ou seja, é a capacidade do computador de aprender.

Trata -se de um processo continuo e evolutivo, cuja a interação com o ambiente através de dados (dataset) , os quais são persistido através de um modelo. Funciona da seguinte forma é recolhido o dataset e desenvolve- se um algoritmo no qual é gerado um modelo que culmina com duas ações que o aprendizado pode ser evoluído e medido.

Do ponto de vista de estrutura de dados, podemos utilizar um classificador de machine learning ou um algoritmo de agrupamento. Nessa estrutura podemos identificar algumas características do negócio: os atributos ou as dimensões.

Fazendo uma correlação com o Tempo teremos: a temperatura, a umidade se têm vento e etc. Um outro ponto importante é a classe consiste num atributo especial em atividades de classificação é tudo aquilo que eu quero descrever ou p rever vai depender do tipo de negócio e da tarefa que está sendo realizada. Também vamos ter a instâncias que do ponto de vista de uma planilha consiste em uma linha onde cada a linha é um dado de negócio coletado.



Nós chamamos de relação um conjunto de dados, dois tipos de dados são importantes para o machine learning.

* Nominais (discreto)
* Númericos (contínuo)

Devido ao campo de aprendizado de máquina oferece uma grande quantidade de opções quando se trata de algoritmos. Como escolher o algoritmo correto? Seria fantástico se houve uma única resposta para tal questionamento, no entanto depende da natureza da tarefa a ser desenvolvida, da quantidade de dados disponíveis, do tipo de resposta esperada, do tempo e do recurso disponível. Apesar de parecer frustrante, esse fato nos leva à uma pergunta muito mais interessante o que devemos saber antes de escolher os algoritmos de aprendizado de máquina para a resolução de um problema?

1. Supervisionado Versus Não Supervisionado

As tarefas de mineração de dados são ditas supervisionadas quando existe uma classe, ou um atributo especial com o qual se pode comparar e validar o resultado. Já o objetivo do um modelo não supervisionado é organizar os dados de alguma forma ou descrever sua estrutura.

Classificação : Supervisionada

Agrupamento, Rega de Associação: Não Supervisionada

1. Classificação

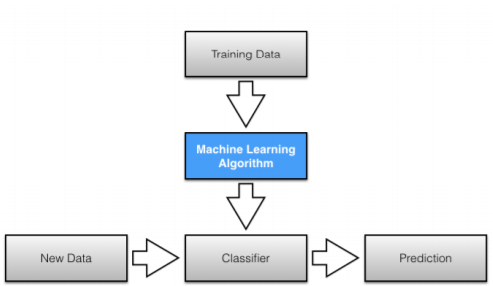
Quando queremos prever ou descrever a classe de um evento. Normalmente a classe em relação esta representada em um atributo especial, posicionada como último atributo da relação. Ou seja são métodos que buscam explicar uma variável categórica, com duas categorias (variável binária) ou mais resumindo é quando não se conhece as possíveis categorias existentes.

* 1. Naive Bayes

Temos esse cabeçalho: Vão influenciar a Classe de forma independente

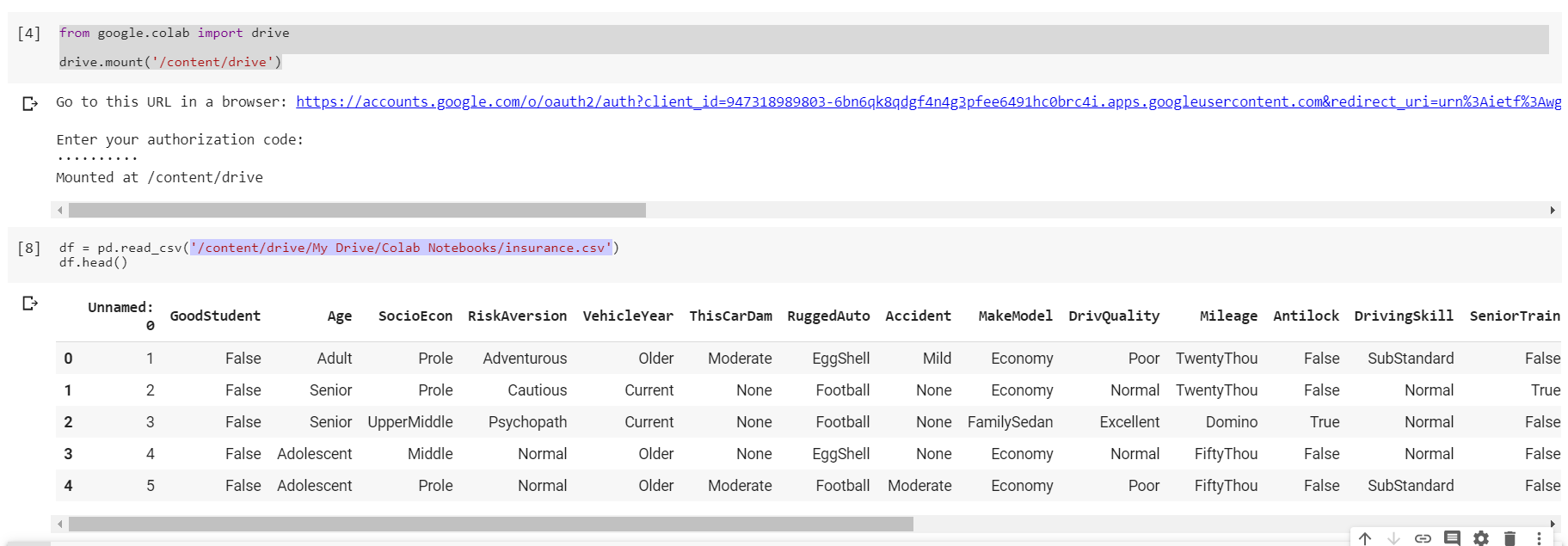
* Queremos saber
* Risco do Seguro
* Probabilidade Condicional da Classe
  + - * 1ª coisa é calcular a probabilidade da classe ( Severe, None, Moderate or Mild)
      * 2ª Verificar a quantidade de instâncias: 19929.

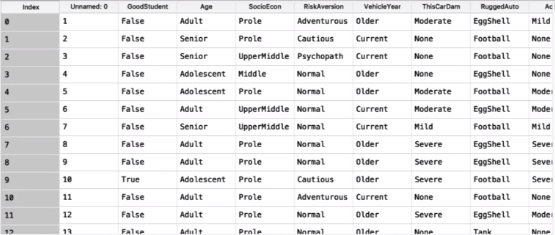
Saber diferenciar qual tarefa se deseja executar e qual o tipo de resposta se deseja obter.



Consiste em classificadores lineares cujo os modelos “ingênuos” baseado no Teorema de Bayes vem da suposição de que os recursos de um conjunto de dados são independentes entre si, ou seja essa suposição de que classificadores lineares conhecidos por serem simples ainda que muito eficiente. O modelo probabilístico de classificadores ingênuos baseia -se no Teorema de Bayes, cuja suposição de que os recursos de um conjunto de dados são independentes entre si. Essa suposição é frequentemente violada, uma vez que estes são robustos, de fácil de implementação, rápido e preciso.

Colab:





A previsão do Risco do Seguro do Veículo, no atributo ACCIDENT ou seja o risco da pessoa sofrer um acidente é médio e nenhum é moderado e alto ou severo. Vamos supor que você queira solicitar o seguro para o seu veículo, você irá passar todos esses dados para a seguradora e esta vai passar no Sistema e indicar qual é o risco de você se envolver em um acidente e qual será o valor do seguro por exemplo.

Explicando o Codigo:

base = pd.read\_csv('/content/drive/My Drive/Colab Notebooks/insurance.csv')

base = base.drop(columns = ['Unnamed: 0'])

base.Accident.unique()

X = base.iloc[:,[0,1,2,3,4,5,6,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26]].values

y = base.iloc[:, 7].values

labelencoder = LabelEncoder()

X[:,0] = labelencoder.fit\_transform(X[:,0])

X[:,1] = labelencoder.fit\_transform(X[:,1])

X[:,2] = labelencoder.fit\_transform(X[:,2])

X[:,3] = labelencoder.fit\_transform(X[:,3])

X[:,4] = labelencoder.fit\_transform(X[:,4])

X[:,5] = labelencoder.fit\_transform(X[:,5])

X[:,6] = labelencoder.fit\_transform(X[:,6])

X[:,7] = labelencoder.fit\_transform(X[:,7])

X[:,8] = labelencoder.fit\_transform(X[:,8])

X[:,9] = labelencoder.fit\_transform(X[:,9])

X[:,10] = labelencoder.fit\_transform(X[:,10])

X[:,11] = labelencoder.fit\_transform(X[:,11])

X[:,12] = labelencoder.fit\_transform(X[:,12])

X[:,13] = labelencoder.fit\_transform(X[:,13])

X[:,14] = labelencoder.fit\_transform(X[:,14])

X[:,15] = labelencoder.fit\_transform(X[:,15])

X[:,16] = labelencoder.fit\_transform(X[:,16])

X[:,17] = labelencoder.fit\_transform(X[:,17])

X[:,18] = labelencoder.fit\_transform(X[:,18])

X[:,19] = labelencoder.fit\_transform(X[:,19])

X[:,20] = labelencoder.fit\_transform(X[:,20])

X[:,21] = labelencoder.fit\_transform(X[:,21])

X[:,22] = labelencoder.fit\_transform(X[:,22])

X[:,23] = labelencoder.fit\_transform(X[:,23])

X[:,24] = labelencoder.fit\_transform(X[:,24])

X[:,25] = labelencoder.fit\_transform(X[:,25])

**X\_treinamento, X\_teste, y\_treinamento, y\_teste = train\_test\_split(X, y,**

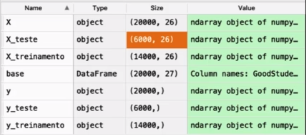
**test\_size = 0.3,**

**random\_state = 0)**

train\_test\_split é aqui que eu realizo a divisão entre base de teste e base de treinamento e utilizar essa função em teste.

test\_size = 0.3, me diz que eu vou utilizar 70% para fazer o treinamento ou seja para gerar a tabela de probabilidade.

random\_state = 0 informa que sempre serão utilizados os mesmos registros para fazer a divisão dessa base de dados.



Note que o x\_treinamento são 14 mil registros e o X lembrando que somente os atributos visores o y\_treinamento 14 mil registros, x\_teste 6 mil e y\_teste 6 mil. Ou seja a tabela de probabilidade vai ser gerada utilizando esses 14 mil registros.

modelo = GaussianNB() /Criação do Modelo

modelo.fit(X\_treinamento, y\_treinamento) / Método que cria a tabela de Probabilidade

previsoes = modelo.predict(X\_teste) / Realizada a Previsão em cada uma das classes e a que tiver o maior valor ele vai assumir como a classe correta prevista pelo algoritmo.

accuracy\_score(y\_teste, previsoes)/ Verifica o quanto ele está acertando

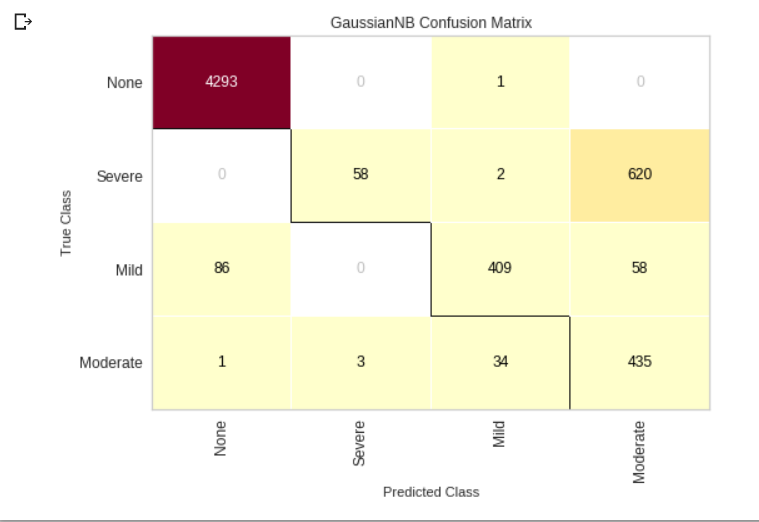
**0.86 ou seja acertou 86% dos registros**

confusao = ConfusionMatrix(modelo, classes=['None','Severe','Mild','Moderate'])/ visualização classe por classe.

confusao.fit(X\_treinamento, y\_treinamento)

confusao.score(X\_teste, y\_teste)

confusao.poof()



* + Em Relação a Classe None:

Temos uma intersecção o risco None, corresponde ao valor de 4 mil 293 na intersecção, ou seja são pessoas que não possuem riscos e foram classificadas corretamente.

Na intersecção do None com o Severe o valor 0 ou seja nenhuma pessoa recebeu tal classificação.

Onde consta o número 1 corresponde a um erro.

* + Em Relação a Classe Severe:

Perceba que ele obteve o total de 58 acertos.

2 registros ele classificou como Mild e 620 como risco Moderate.

* + Em Relação a Classe Mild:

Temos 409 corretos.

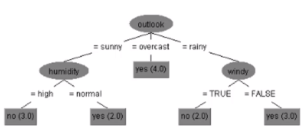
* + Em Relação a Classe Moderate:

Ele acertou 435 e errou 34.

Baseando -se na análise acima, chegamos a conclusão que a Classe Severe possui os piores acertos.

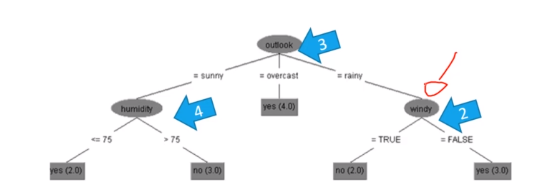
* 1. Arvore de Decisão

Consiste em métodos de aprendizado de máquinas supervisionado não-paramétricos, muito utilizados em tarefas de classificação e regressão. De um modo geral em computação, são estruturas de dados formadas por um conjunto de elementos que armazenam informações chamadas nós. Como toda árvore possui um nó chamado raiz, que possui o maior nível hierárquico e ligações para outros elementos, denominados filhos. Em uma árvore de decisão, uma decisão é tomada através do caminhamento a partir do nó raiz até o nó folha.



O processo de classificação ocorre do seguinte modo, a arvore é induzida, ou seja, ela é criada e torna -se um modelo. Ou seja, se tem simplesmente um único Nó raiz, o qual você vai olhar nos dados que está buscando a classificação, vai verificar em cada atributo seu valor e vai percorrer até chegar ao nó folha.

Indução da Árvore



1. Em duas partes, ou em n partes.
2. Binários: divisão dupla.
3. Nominal: múltiplas divisões.
4. Continuo: comparação de valores ou desratização

Divisão de Árvores

Objetivando é criar divisões puras possíveis, ou seja os valores de um determinado atributo leva a uma única classe. Para se escolher qual atributo vai criar uma divisão se aplica um algoritmo buscando ver qual tem a maior ou melhor pureza.

* Gini
* Entropia
* Erro de Classificação

Condição de Parada

É quando um atributo tem divisões em valores e uma determinada divisão por exemplo Sunny ou seja leva apenas a classe Sim, obvio que termos uma condição de parada porque não tenho como dividir mais, no entanto nem sempre se chega ao nó da folha, daí utilizamos outro critério que corresponde ao número mínimo de observações em um nó.

Poda

É a redução do tamanho da árvore.

No Colab:

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.tree import export\_graphviz

from sklearn.preprocessing import LabelEncoder

from sklearn.metrics import accuracy\_score

from yellowbrick.classifier import ConfusionMatrix

base = pd.read\_csv('/content/drive/My Drive/Colab Notebooks/credit-g.csv')/Carregando a base de dados



Verificar se eu vou ou não conseguir um empréstimo para determinada pessoa.

X = base.iloc[:,0:20].values

y = base.iloc[:, 20].values



labelencoder = LabelEncoder() /Atributos Categóricos

X[:,0] = labelencoder.fit\_transform(X[:,0])

X[:,2] = labelencoder.fit\_transform(X[:,2])

X[:,3] = labelencoder.fit\_transform(X[:,3])

X[:,5] = labelencoder.fit\_transform(X[:,5])

X[:,6] = labelencoder.fit\_transform(X[:,6])

X[:,8] = labelencoder.fit\_transform(X[:,8])

X[:,9] = labelencoder.fit\_transform(X[:,9])

X[:,11] = labelencoder.fit\_transform(X[:,11])

X[:,13] = labelencoder.fit\_transform(X[:,13])

X[:,14] = labelencoder.fit\_transform(X[:,14])

X[:,16] = labelencoder.fit\_transform(X[:,16])

X[:,18] = labelencoder.fit\_transform(X[:,18])

X[:,19] = labelencoder.fit\_transform(X[:,19])

X\_treinamento, X\_teste, y\_treinamento, y\_teste = train\_test\_split(X, y,

test\_size = 0.3,

random\_state = 0)

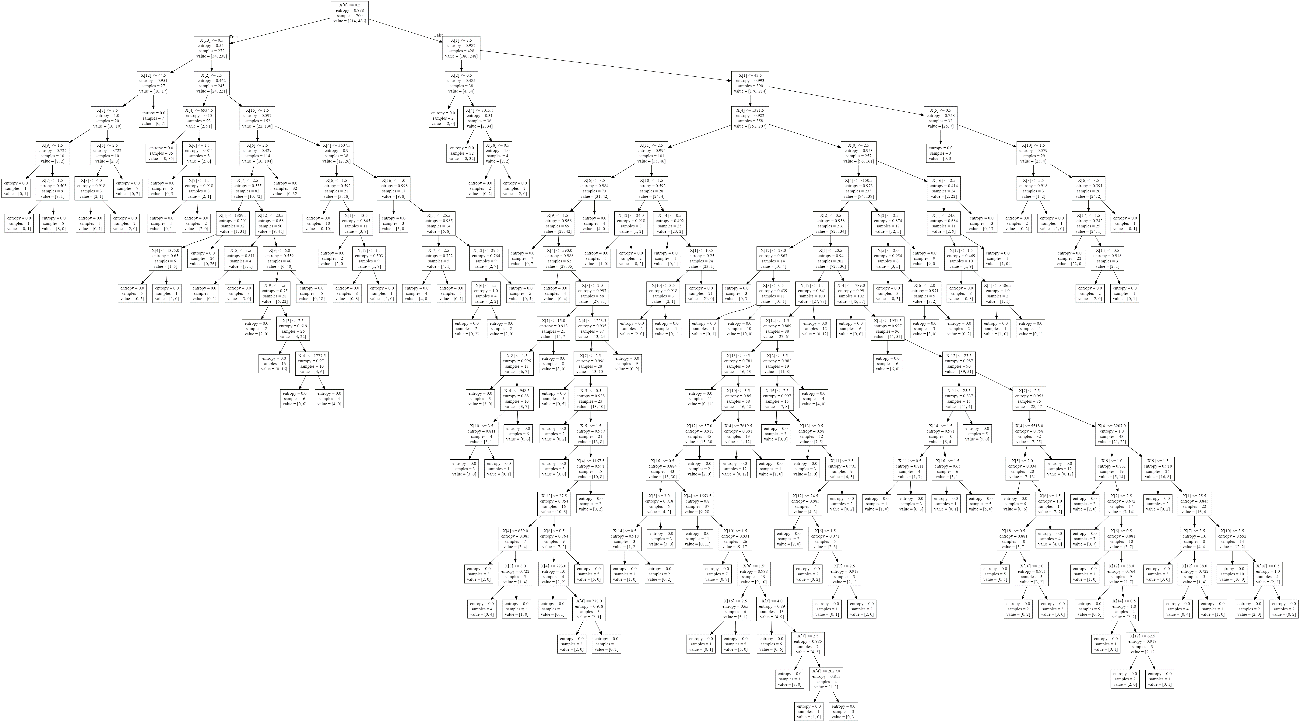
/usamos três modleos diferentes, com parametrizações diferentes

# Modelo 1

modelo1 = DecisionTreeClassifier(criterion = 'entropy') /ganho de informação

modelo1.fit(X\_treinamento, y\_treinamento) /700 registros

export\_graphviz(modelo1, out\_file = 'modelo1.dot') /modelo gerado



previsoes1 = modelo1.predict(X\_teste)

accuracy\_score(y\_teste, previsoes1)

confusao1 = ConfusionMatrix(modelo1)

confusao1.fit(X\_treinamento, y\_treinamento)

confusao1.score(X\_teste, y\_teste)

confusao1.poof()



# Modelo 2

modelo2 = DecisionTreeClassifier(criterion = 'entropy', min\_samples\_split = 20)

modelo2.fit(X\_treinamento, y\_treinamento)

export\_graphviz(modelo2, out\_file = 'modelo2.dot')

previsoes2 = modelo2.predict(X\_teste)

accuracy\_score(y\_teste, previsoes2)

confusao2 = ConfusionMatrix(modelo2)

confusao2.fit(X\_treinamento, y\_treinamento)

confusao2.score(X\_teste, y\_teste)

confusao2.poof()

# Modelo 3

modelo3 = DecisionTreeClassifier(criterion = 'entropy', min\_samples\_leaf = 5, min\_samples\_split = 20)

modelo3.fit(X\_treinamento, y\_treinamento)

export\_graphviz(modelo3, out\_file = 'modelo3.dot')

previsoes3 = modelo3.predict(X\_teste)

accuracy\_score(y\_teste, previsoes3)

confusao3 = ConfusionMatrix(modelo3)

confusao3.fit(X\_treinamento, y\_treinamento)

confusao3.score(X\_teste, y\_teste)

confusao3.poof()

Conclusão

Um algoritmo convencional é baseado em uma entrada que consiste nos dados atuais, ou seja você precisa dos dados da rota, já no machine elarning eu preciso de dados históricos para construir um modelo ou seja os dados históricos vão ser submetidos ao algoritimo e ele vai gerar como saída um modelo o qual será utilizado para fazer as previsões.Uma vez criado, não se faz mais necessário os dados históricos nem do algoritmo apensa do modelo.

Nos modelos supervisionados o procedimento mais comum para realizar uma análise de dados é dividir o conjunto de dados em duas partes: treino e teste. O modelo de aprendizado de máquina é então treinado no dataset de treino, logicamente. Posteriormente o modelo é aplicado para realizar predições no conjunto de dados de teste, momento no qual é feita uma avaliação da qualidade das predições.

Já nos modelos não supervisionados a abordagem é diferente. Isso ocorre porque como não há uma variável específica a ser explicada (ou seja, não há um target), então não há sentido em treinar o conjunto de dados, pois também não será possível avaliar a assertividade do modelo. No aprendizado de máquina não superviosionado procuramos encontrar padrões, perfis, itens semelhantes.