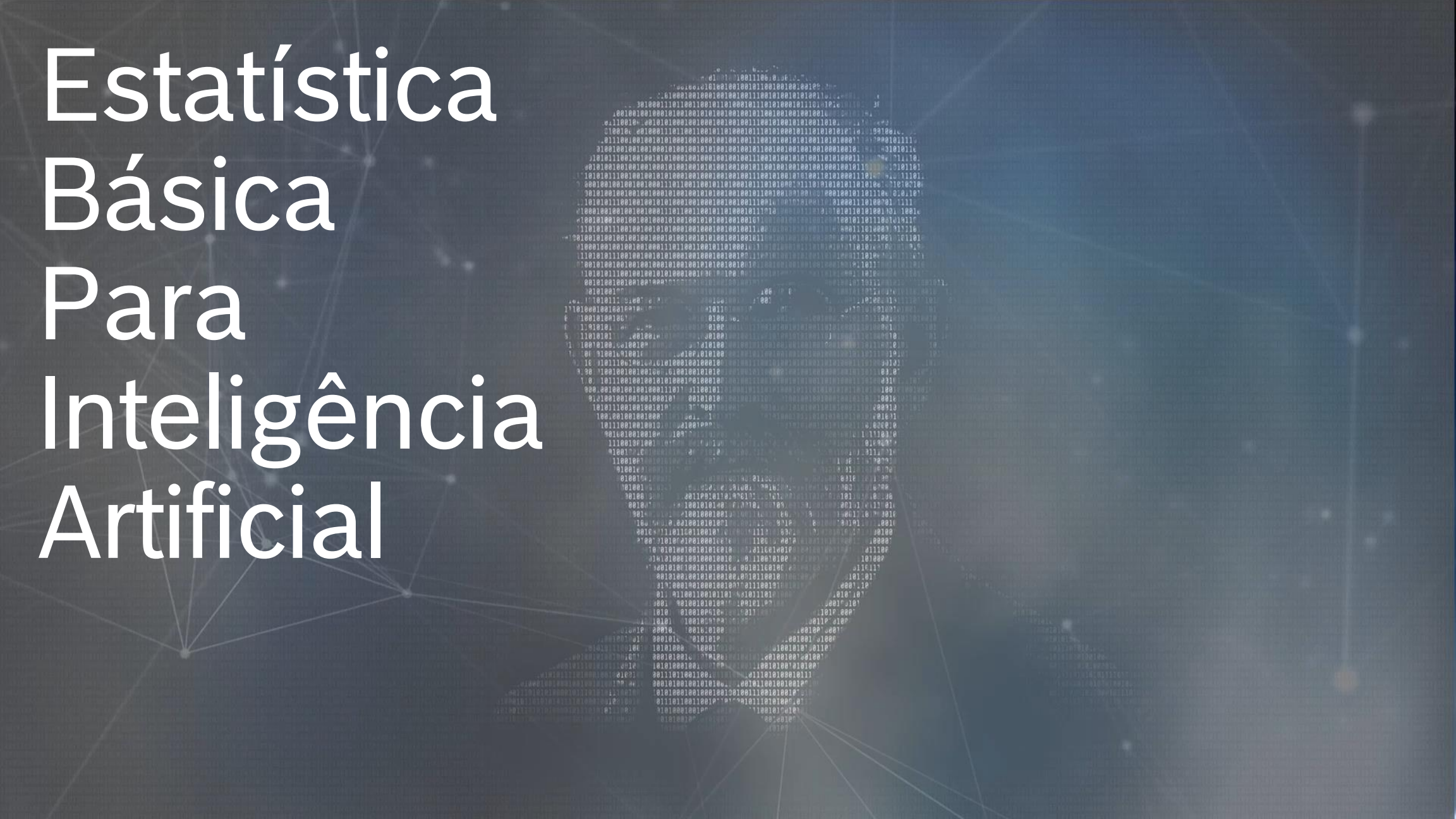
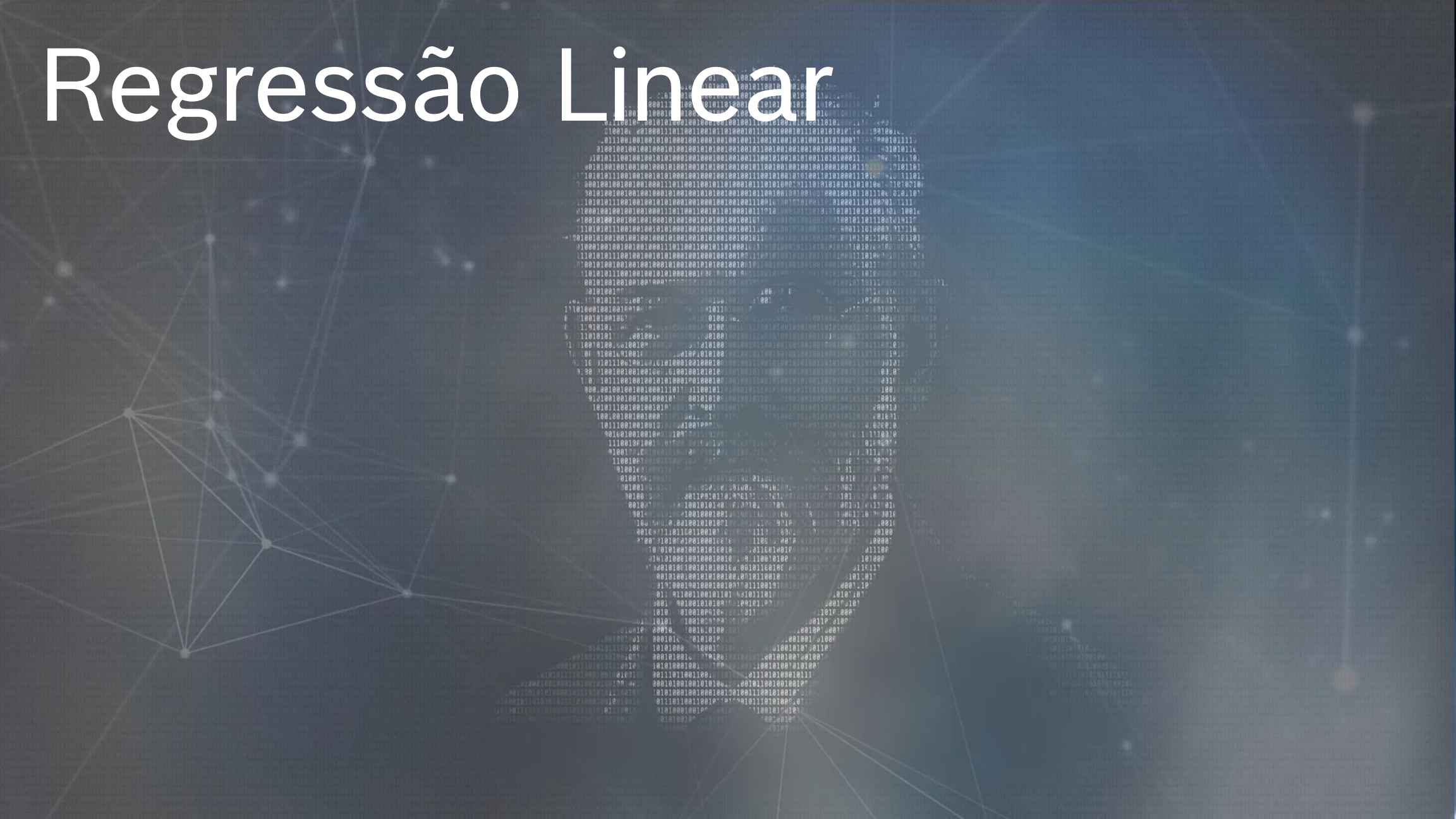
Estatística Básica Para Inteligência Artificial



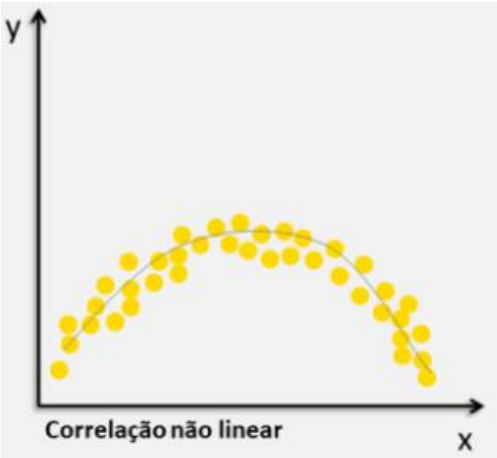
Regressão Linear



Regressão Linear
O que é Regressão Linear

Sala Digital

- Regressão linear simples é um modelo de equação matemática que inclui duas variáveis e apresenta uma linha entre elas;
- Serve para prever comportamentos com base na associação entre essas duas variáveis, que geralmente possuem uma correlação;
- Através de um gráfico de dispersão é mais fácil observar a linearidade. Teremos duas variáveis.
- Ex: idade x tempo de estudo.



BOSCH

Regressão Linear
Variáveis Numéricas

Sala Digital

- Em uma regressão linear simples as duas variáveis são chamadas de independente e dependente. Isso significa que a variável dependente é a que será explicada e a independente é que será usada para explicar a variação (comportamento) da dependente;

Resultado → Causa

Dependente	Independente
Nota da prova	Tempo de estudo
Tempo de uso de ar-condicionado	Sensação térmica na cidade
Frequência de compras online	Valor do frete

Regressão Linear
Variáveis numéricas

Sala Digital

- A regressão linear simples funciona para dados contínuos, ou seja, dados com valores numéricos, dados quantitativos, que representam medidas;
- Caso o dado seja qualitativo, deve-se transformá-los em variáveis numéricas.

Refere a qualidade, geralmente é um adjetivo.

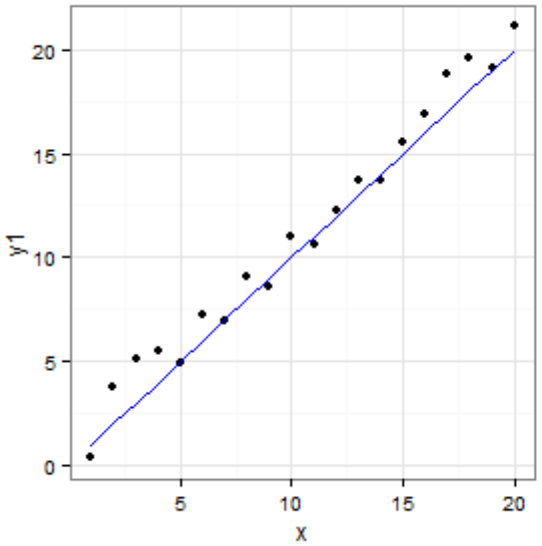
Qualitativo	Quantitativo
Sexo (homem, mulher)	Homem = 0 Mulher = 1
Cor dos olhos (castanho, verde, azul)	Castanho = 0 Verde = 1 Azul = 2

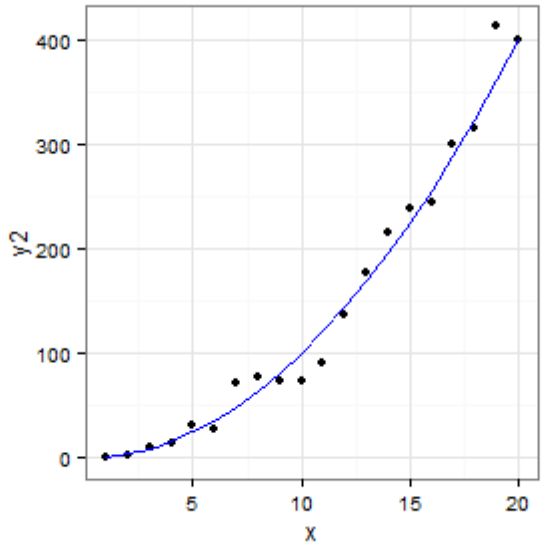
Refere a quantidade, dado representado por um número.

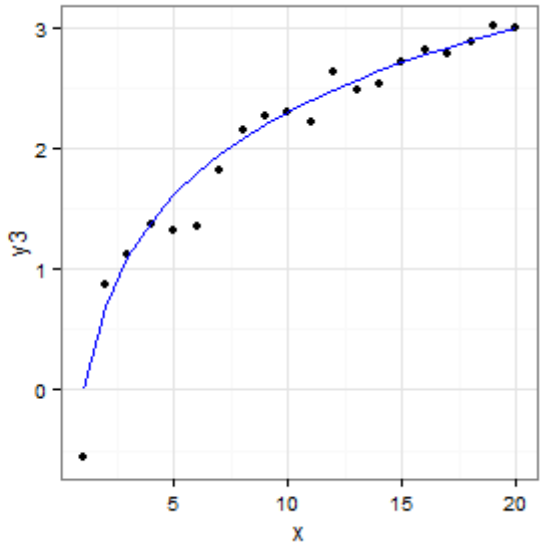
Regressão Linear
Parâmetros de Regressão

Sala Digital

- São as variáveis que formam a equação linear que descreve a relação entre as variáveis;
- O termo linear é usado para indicar que o modelo é linear nos parâmetros da regressão.
- Y=f(x).







BOSCH

Regressão Linear
Parâmetros da Regressão

Sala Digital

- A equação que descreve a regressão, é formada pelas variáveis dependente, independente, constante e o coeficiente.
- Juntas elas formam uma equação de primeiro grau que descreve a reta que melhor aproxima os dados.

$$\boldsymbol{y = b_0 + b_1 * x_1}$$

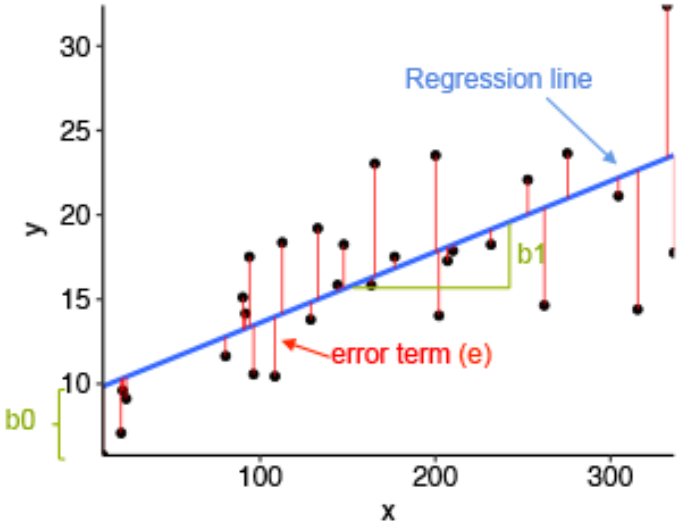
y = Dependente
b_0 = Constante
b_1 = Coeficiente
x_1 = Independente

- Por exemplo, queremos saber qual seria o custo do plano de saúde de acordo com a idade da pessoa. Nesse caso teremos:
  - Y sendo custo do plano de saúde;
  - B0 e B1 os valores que determinam a reta, e iremos calcular utilizando Python para encontrar a melhor posição da reta. B0 indica onde a reta começa e o B1 indica a inclinação da reta;
  - X1 sendo a idade da pessoa.

Regressão Linear
Aproximação e Erro

Sala Digital

- Para avaliar o modelo de regressão, precisamos calcular a distância dos pontos (dados) até a reta. Essa distância é o erro entre o valor previsto (reta) e o valor real; a intenção é reduzir ao máximo esse erro. A reta passa no “meio” dos pontos.
- A técnica mais utilizada é o mean squared erro (MSE, ou Erro quadrático Médio);
- Lembrando que é necessário eliminar os outliers.



- A fórmula do MSE é:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(f_i - y_i)^2$$

- Onde:
  - N = Total de amostras;
  - Fi = Valor calculado pelo modelo;
  - Yi = Preço real.

BOSCH

Regressão Linear
Aproximação e Erro

Sala Digital

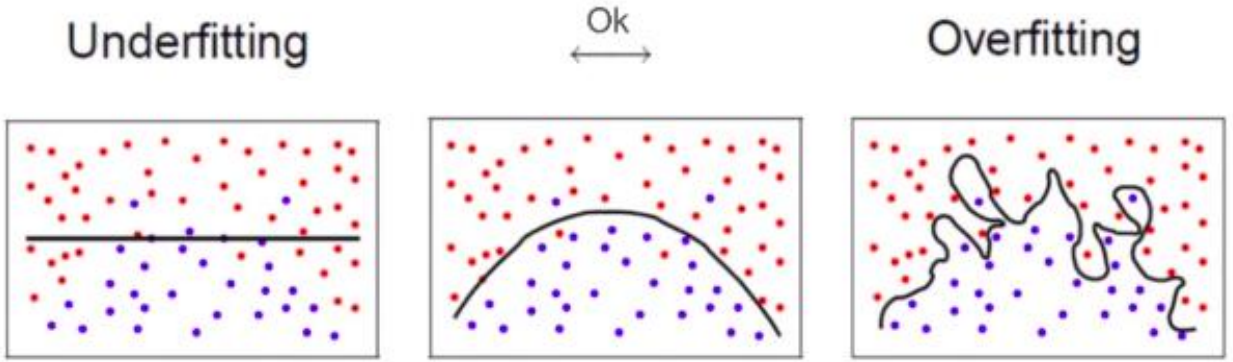
- Olhando para fórmula, parece um pouco complicado. Vamos descomplicar o entendimento;
- O MSE penaliza os erros maiores;
- Quanto menor o valor de MSE, mais precisa a reta vai estar;
- Fazendo o treinamento do modelo para encontrar parâmetros de B0 e B1 que minimizem o erro.

Preço real (y)	Preço calculado (f)	Erro
150	180	$(150 - 180)^2 = 900$
60	55	$(60 - 55)^2 = 25$
220	230	$(220 - 230)^2 = 100$
45	67	$(45 - 67)^2 = 484$
	Somando os valores de erro e dividindo pelo total de amostras.	1509/4 = 377,25

Regressão Linear
Viés e Variância

Sala Digital

- Um valor de Viés alto, significa que o modelo não está aprendendo como deveria. Já um valor muito baixo indica que o modelo está se adaptando muito aos dados de treinamento, o que poderá gerar muito erro com novos dados;
- A variância indica a “sensibilidade” do modelo a novos dados. Uma variância alta nos levará a um modelo super adaptado aos dados de treinamento;
- Viés está relacionado ao modelo se ajustar aos dados (underfitting), e a variância está relacionada com o modelo se ajustar a novos dados (overfitting).



BOSCH

Regressão Linear
Exemplo

Sala Digital

- Com base nos dados de idade e valor do plano de saúde, vamos construir um modelo de regressão linear para que possamos prever qual será o valor do plano de acordo com a idade da pessoa.

```
#Importando as bibliotecas
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
```

```
#Criando a base dados
dados = {'Valor':[200, 220, 300, 290, 450, 457, 500, 530, 700, 800],
         'Idade': [18, 22, 23, 30, 35, 44, 49, 50, 67, 75]}
dados = pd.DataFrame(data=dados)
```

```
#Separando os dados
#X é a variável independente
#Y é a variavel dependente
X = dados['Idade'].values
Y = dados['Valor'].values
```

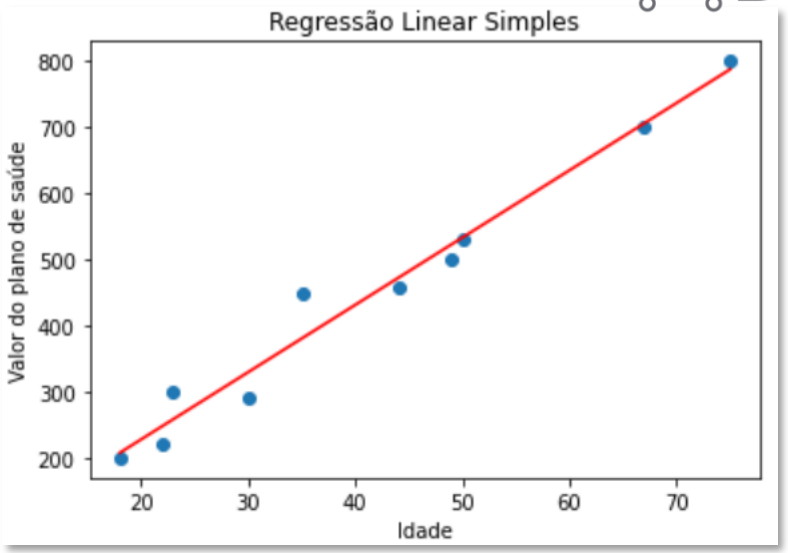
Regressão Linear
Exemplo

Sala Digital

```
#Função para usar X transposto
X = X.reshape(-1, 1)
```

```
#Definindo o regressor linear
regressor = LinearRegression()
#Passando os dados para
#treinar o regressor
regressor.fit(X, Y)
```

```
#Visualizando o gráfico
plt.scatter(X, Y)
plt.plot(X, regressor.predict(X), color='red')
plt.title('Regressão Linear Simples')
plt.xlabel('Idade')
plt.ylabel('Valor do plano de saúde')
```




```
#Prevendo novos valores
idade = np.array(57)
previsao1 = regressor.predict(idade.reshape(-1,1))
previsao2 = regressor.intercept_ + regressor.coef_*idade
print(previsao1)
print(previsao2)
```

```
[604.39408838]
[604.39408838]
```

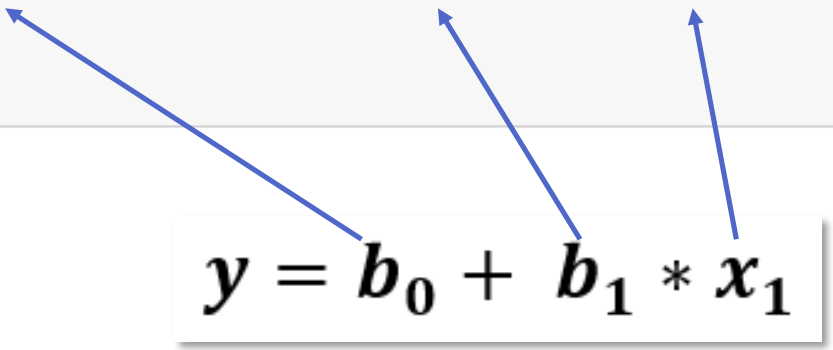

BOSCH

Regressão Linear
Exemplo

Sala Digital

```
#Prevendo novos valores
idade = np.array(57)
previsao1 = regressor.predict(idade.reshape(-1,1))
previsao2 = regressor.intercept_ + regressor.coef_*idade
print(previsao1)
print(previsao2)
```

```
[604.39408838]
[604.39408838]
```



$$y = b_0 + b_1 * x_1$$

BOSCH

Regressão Linear
Exercício

Sala Digital

- O valor de um imóvel geralmente é calculado com base na sua área, quanto maior a área, mais caro o imóvel tende a ser. Isso nos indica que essa relação, é uma relação linear. Com base nisso, construa o gráfico que mostre a reta de regressão e verifique se existe algum outlier nos dados fornecidos. Utilize o regressor linear para prever também os seguintes valores dos imóveis com as seguintes áreas: 35, 70, 190.
- Tempo estimado: 60 min.

Área	40	45	50	53	60	65	100	110	113	130
Valor	120	180	190	187	195	200	300	320	305	400

Regressão Linear
Exercício - Resposta

Sala Digital

```
#Importando as bibliotecas
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
```

```
#Criando nosso dataset
dados = {'Area':[40, 45, 50, 53, 60, 65, 100, 110, 113, 130],
         'Valor':[120, 180, 190, 187, 195, 200, 300, 320, 305, 400]}
dados = pd.DataFrame(data=dados)
```

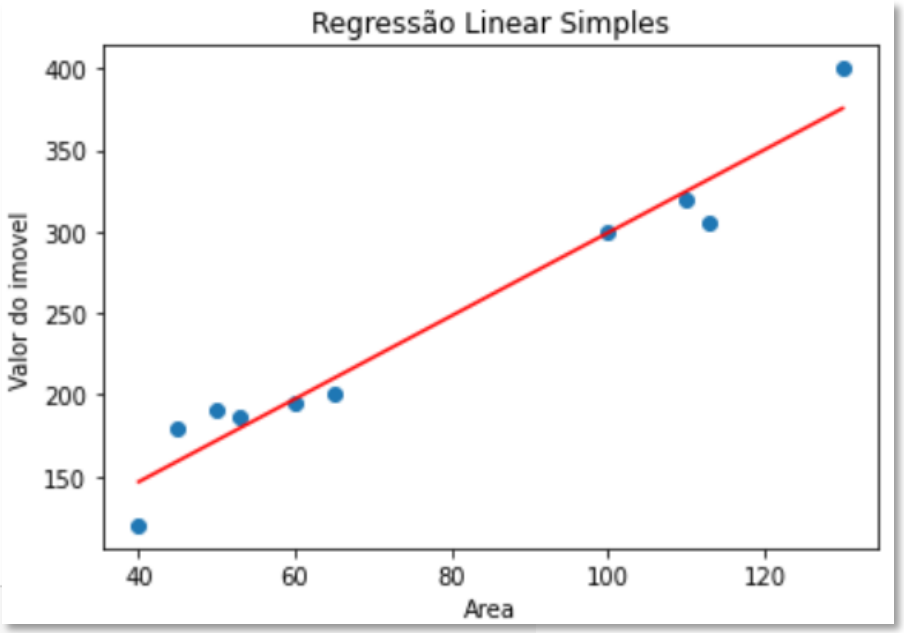
```
#Separando as variáveis
independente = dados['Area'].values
dependente = dados['Valor'].values
```

```
#Transpondo os valores
independente = independente.reshape(-1,1)
```

Regressão Linear
Exercício - Resposta

Sala Digital

```
#Criando o regressor
regressor = LinearRegression()
#Passando os dados para
#treinar o regressor
regressor.fit(independente, dependente)
```



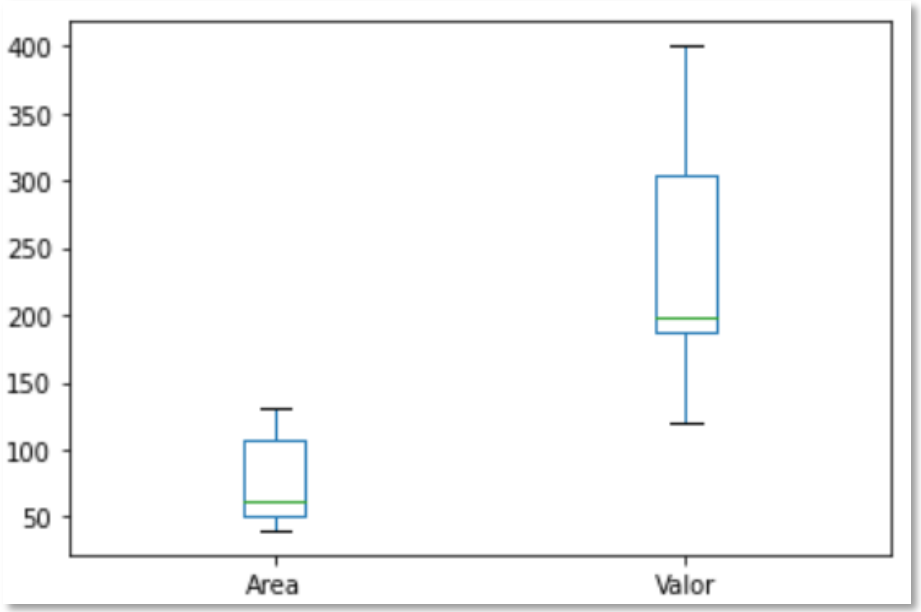

```
#Visualizando o gráfico da regressão
plt.scatter(independente, dependente)
plt.plot(independente, regressor.predict(independente), color='red')
plt.title('Regressão Linear Simples')
plt.xlabel('Area')
plt.ylabel('Valor do imovel')
```


BOSCH

Regressão Linear
Exercício - Resposta

Sala Digital

```
#Verificando se há outliers
#Gerando um gráfico de boxplot
dados.boxplot(column=['Area', 'Valor'], grid=False)
```



BOSCH

Regressão Linear
Exercício - Resposta

Sala Digital

```
#Usando o regressor para
#prever novos valores
area = np.array(35)
previsao = regressor.predict(area.reshape(-1,1))
print("O valor previsto é: ", previsao)
```

```
O valor previsto é:  [133.86413463]
```

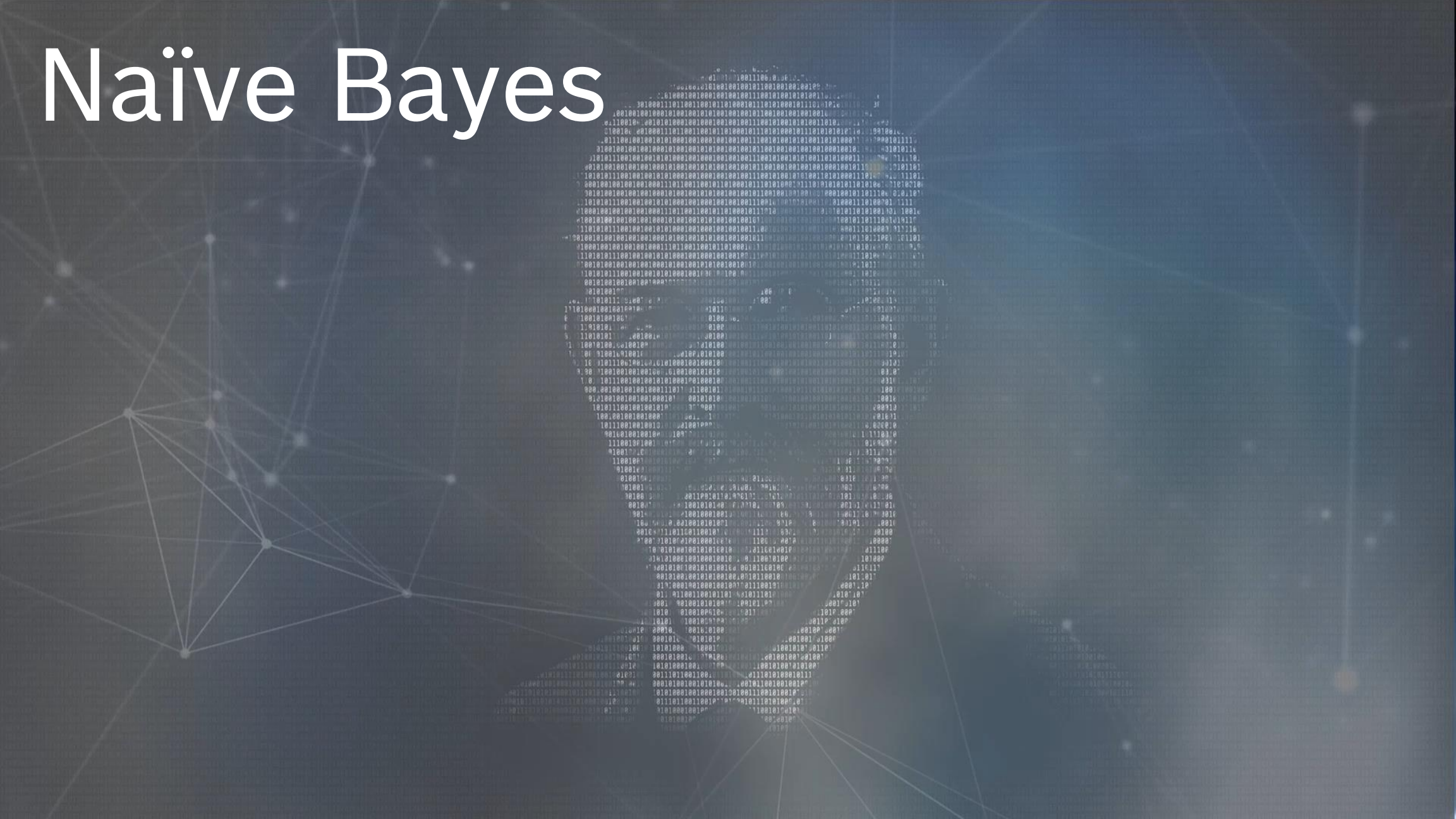
```
#Usando o regressor para
#prever novos valores
area = np.array(190)
previsao = regressor.predict(area.reshape(-1,1))
print("O valor previsto é: ", previsao)
```

```
O valor previsto é:  [528.2044984]
```

```
#Usando o regressor para
#prever novos valores
area = np.array(70)
previsao = regressor.predict(area.reshape(-1,1))
print("O valor previsto é: ", previsao)
```

```
O valor previsto é:  [222.9087329]
```

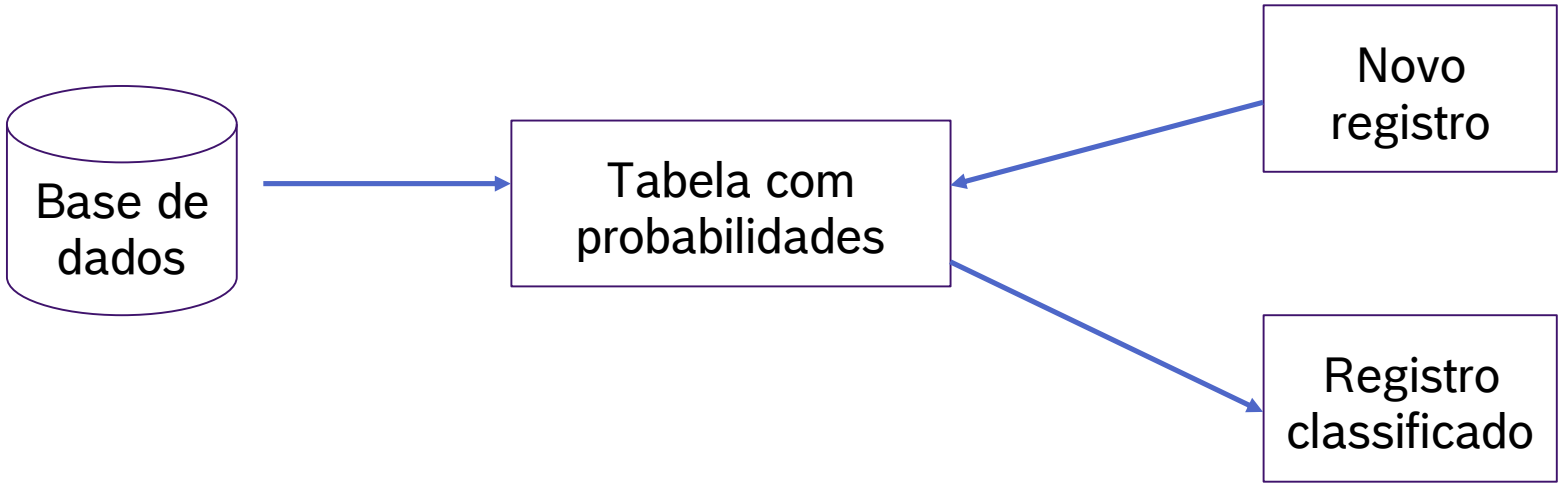
Naïve Bayes



Naïve Bayes
Teoria e Conceitos

Sala Digital

- Possui uma abordagem probabilística, pois é baseado no Teorema de Bayes;
- O algoritmo Naive Bayes é um algoritmo de classificação de dados;
- O treinamento se baseia em utilizados os dados de treinamento e calcular as probabilidades dos atributos previsores;
- Quando um novo registro é submetido, o retorno do algoritmo será a probabilidade desse registro pertencer a determinada classe.



Naïve Bayes
Utilizações

Sala Digital

- Principais utilizações:
  - Classificações de textos de notícias ou artigos;
  - Filtragem de spam no e-mail;
  - Analise de sentimento em textos;
  - Sistema de recomendação;
  - Previsões com muitas classes;
  - Medicina de diagnósticos;
  - Previsão do tempo.

BOSCH

Naïve Bayes
Problemática

Sala Digital

- Algumas problemáticas e bases de dados não funcionam com o método de regressão linear;
- Problemática: Queremos prever em qual dia determinado time irá treinar;
- Dados:
  - Histórico do tempo(clima);
  - Umidade do ar;
  - Vento;
  - Houve treino?
- Com base nesse dados, podemos calcular a probabilidade do time treinar ou não em dias futuros utilizando o algoritmo Naive Bayes;
- Vamos ver a tabela dos dados a seguir para entender como funciona.

Naïve Bayes
Problemática: Resolvendo

Sala Digital

	Atributos previsores			Classe
Dia	**Tempo**	**Humidade**	**Vento**	**Treinou?**
D1	Ensolarado	Alta	Fraco	Não
D2	Ensolarado	Alta	Forte	Não
D3	Nublado	Alta	Fraco	Sim
D4	Chuvoso	Alta	Fraco	Sim
D5	Chuvoso	Normal	Fraco	Sim
D6	Chuvoso	Normal	Forte	Não
D7	Nublado	Normal	Forte	Sim
D8	Ensolarado	Alta	Fraco	Não
D9	Ensolarado	Normal	Fraco	Sim
D10	Chuvoso	Normal	Fraco	Sim
D11	Ensolarado	Normal	Forte	Sim
D12	Nublado	Alta	Forte	Sim
D13	Nublado	Normal	Fraco	Sim
D14	Chuvoso	Alta	Forte	Não

Naïve Bayes
Problemática: Resolvendo

Sala Digital

- Primeiro devemos construir as tabelas de **frequência** para cada uma das colunas:

Tabela de frequência		Treinou?	
		Sim	Não
Vento	Forte	6	2
	Fraco	3	3

Tabela de frequência		Treinou?	
		Sim	Não
Humidade	Alta	3	4
	Normal	6	1

Tabela de frequência		Treinou?	
		Sim	Não
Tempo	Ensolarado	3	2
	Nublado	4	0
	Chuvoso	3	2

Naïve Bayes
Problemática: Resolvendo

Sala Digital

- Para cada tabela de frequência, devemos construir a tabela de **probabilidade**:

Tabela de probabilidade		**Treinou?**		
		Sim	Não	
Vento	Forte	3/9	3/5	6/14
	Fraco	6/9	2/5	8/14
		9/14	5/14	

Tabela de probabilidade		**Treinou?**		
		Sim	Não	
Humidade	Alta	3/9	4/5	7/14
	Normal	6/9	1/5	7/14
		9/14	5/14	

Naïve Bayes
Problemática: Resolvendo

Sala Digital

- Explicando a tabela:

P(x|c) = P(Ensolarado|Sim) = 2/9 = 0.22
Probabilidade de treino sendo sol

Tabela de probabilidade		Treinou?		
		Sim	Não	
Tempo	Ensolarado	2/9	3/5	5/14
	Nublado	4/9	0/5	4/14
	Chuvoso	3/9	2/5	5/14
		9/14	5/14	

P(x) = P(Ensolarado) = 5/14 = 0.36
Probabilidade de estar ensolarado

P(x) = P(Sim) = 9/14 = 0.64
Probabilidade de haver treino

Naïve Bayes
Prevendo Novos Dados

Sala Digital

- Vamos prever se haverá treino ou não com base em:
  - Tempo = Chuvoso;
  - Umidade = Alta;
  - Vento = Fraco;
- Probabilidade de ‘Sim’ neste dia:
  - $P(Tempo = Chuvoso|Sim) * P(Umidade = Alta|Sim) * P(Vento = Fraco|Sim) * P(Sim)$
  - $= 3/9 * 3/9 * 6/9 * 9/14 = 0{,}0476$
- Probabilidade de ‘Não’ neste dia:
  - $P(Tempo = Chuvoso|Não) * P(Umidade = Alta|Não) * P(Vento = Fraco|Não) * P(Não)$
  - $= 2/5 * 4/5 * 2/5 * 5/14 = 0.0166$
- Normalizando os valores:
  - $P(Sim) = \frac{0.0199}{(0.0199 + 0.0476)} = 0{,}74 \qquad P(Não) = \frac{0.0166}{(0.0199 + 0.0476)} = 0.26$

Naïve Bayes
Exemplo Prático com Python

Sala Digital

- Determinado banco possui os dados de histórico de empréstimo, vistos na tabela abaixo. Com esses dados, o banco solicitou que fosse construído um modelo que fornecendo os dados de entrada, indique se deverá fornecer ou não o empréstimo.

Renda	Idade	Empréstimo	Emprestou?
Alta	Jovem	Alto	Sim
Média	Idoso	Alto	Não
Média	Adulto	Médio	Não
Baixa	Adulto	Médio	Não
Baixa	Adulto	Médio	Não
Baixa	Idoso	Baixo	Sim
Baixa	Jovem	Alto	Não
Alta	Jovem	Médio	Sim
Baixa	Jovem	Baixo	Sim
Média	Jovem	Baixo	Sim

Naïve Bayes
Exemplo Prático com Python

Sala Digital

```
#Importando as bibliotecas
import pandas as pd
from sklearn.naive_bayes import GaussianNB
from sklearn import preprocessing
```

```
#Gerando o dataset
dados = {'Renda': ['Alta','Média','Média','Baixa','Baixa','Baixa','Baixa','Alta','Baixa','Média'],
         'Idade':['Jovem','Idoso','Adulto','Adulto','Adulto','Idoso','Jovem','Jovem','Jovem','Jovem'],
         'Valor_Empréstimo':['Alto','Alto','Médio','Médio','Médio','Baixo','Alto','Médio','Baixo','Baixo'],
         'Emprestou':['Sim','Não','Não','Não','Não','Sim','Não','Sim','Sim','Sim']}

dados = pd.DataFrame(data=dados)
```

Naïve Bayes
Exemplo Prático com Python

Sala Digital

```
#Criando o LabelEncoder
renda_lbencoder = preprocessing.LabelEncoder()
idade_lbencoder = preprocessing.LabelEncoder()
valor_lbencoder = preprocessing.LabelEncoder()
emprestou_lbencoder = preprocessing.LabelEncoder()
```

```
#Usando o LabelEncoder para
#atribiur números as variáveis qualitativas
renda_lbencoder.fit(dados['Renda'].unique())
idade_lbencoder.fit(dados['Idade'].unique())
valor_lbencoder.fit(dados['Valor_Empréstimo'].unique())
emprestou_lbencoder.fit(dados['Emprestou'].unique())
```

- LabelEncoder pode transformar [cachorro, gato, cachorro, rato, gato] em [1,2,1,3,2], ou seja, ele codifica os dados apresentados em um dataFrame.

- Fitting é igual a training, depois de treinado o módulo é capaz de fazer estimativas.
- Fitting seu modelo para treinar o dado é especialmente a parte do treinamento de modelagem de processo.
- Por exemplo, ele encontra o coeficiente para uma equação especificada via o algoritmo que foi usado.

Naïve Bayes

Sala Digital

```
#Tranformando o dataset de variáveis qualitativas
#para variáveis quantitativas
dados['Renda'] = renda_lbencoder.transform(dados['Renda'])
dados['Idade'] = idade_lbencoder.transform(dados['Idade'])
dados['Valor_Empréstimo'] = valor_lbencoder.transform(dados['Valor_Empréstimo'])
dados['Emprestou'] = emprestou_lbencoder.transform(dados['Emprestou'])
```

```
#Separando o nosso data set nos atributos previsores
#e na classe objetivo
previsor = dados[['Renda', 'Idade', 'Valor_Empréstimo']]
classe = dados['Emprestou']
```

```
#Criando o classificado NaiveBayes
gnb = GaussianNB()
gnb.fit(previsor, classe)
```

```
#Verificando a precisão
print("Precisão =",gnb.score(previsor, classe)*100,"%")
```

```
Precisão = 80.0 %
```

Naïve Bayes
Exemplo Prático com Python

Sala Digital

```
#Inserindo novos dados para serem previstos
previsao = {'Renda':['Média','Alta'], 'Idade':['Jovem', 'Jovem'], 'Valor_Empréstimo':['Baixo','Alto']}
previsao = pd.DataFrame(data=previsao)

previsao['Renda'] = renda_lbencoder.transform(previsao['Renda'])
previsao['Idade'] = idade_lbencoder.transform(previsao['Idade'])
previsao['Valor_Empréstimo'] = valor_lbencoder.transform(previsao['Valor_Empréstimo'])
```

```
#Verificando o resultado
print(gnb.predict(previsao))
print(emprestou_lbencoder.inverse_transform(gnb.predict(previsao)))
```

```
[1 1]
['Sim' 'Sim']
```

```
#Verificando as probabilidades
gnb.predict_proba(previsao)
```

```
array([[0.16817714, 0.83182286],
       [0.00590879, 0.99409121]])
```

Naïve Bayes
Exercício

Sala Digital

- Para este exercício, vamos utilizar os 10 primeiros dias da tabela utilizada na problemática (que será mostrada no próximo slide), para saber se o time irá treinar ou não. Utilizando Python, construa:
  - O dataset contendo dados;
  - Se necessário, transforme as variáveis qualitativas em quantitativas;
  - O modelo Naive Bayes;
  - Mostre a precisão do modelo;
  - Utilizando o modelo Naive Bayes criado, preveja os dados a seguir.
- Tempo estimado: 60 min.

Naïve Bayes
Exercício

Sala Digital

Dados para treinamento			
Tempo	**Humidade**	**Vento**	**Treinou?**
Ensolarado	Alta	Fraco	Não
Ensolarado	Alta	Forte	Não
Nublado	Alta	Fraco	Sim
Chuvoso	Alta	Fraco	Sim
Chuvoso	Normal	Fraco	Sim
Chuvoso	Normal	Forte	Não
Nublado	Normal	Forte	Sim
Ensolarado	Alta	Fraco	Não
Ensolarado	Normal	Fraco	Sim
Chuvoso	Normal	Fraco	Sim

Dados para prever		
Tempo	**Humidade**	**Vento**
Ensolarado	Normal	Forte
Nublado	Alta	Forte
Nublado	Normal	Fraco
Chuvoso	Alta	Forte

Naïve Bayes
Exercício - Resposta

Sala Digital

```
#Importando as bibliotecas
import pandas as pd
from sklearn.naive_bayes import GaussianNB
from sklearn import preprocessing
```

```
#Gerando o dataset
dados = {'Tempo': ['Ensolarado', 'Ensolarado', 'Nublado', 'Chuvoso', 'Chuvoso', 'Chuvoso',
                   'Nublado', 'Ensolarado', 'Ensolarado', 'Chuvoso'],
         'Umidade':['Alta', 'Alta', 'Alta', 'Alta', 'Normal', 'Normal', 'Normal', 'Alta',
                    'Normal', 'Normal'],
         'Vento':['Fraco', 'Forte', 'Fraco', 'Fraco', 'Fraco', 'Forte', 'Forte', 'Fraco', 'Fraco', 'Fraco',],
         'Treinou':['Não', 'Não', 'Sim', 'Sim', 'Sim', 'Não', 'Sim', 'Não', 'Sim', 'Sim']}

dados = pd.DataFrame(data=dados)
```

```
#Criando o LabelEncoder
tempo_lbencoder = preprocessing.LabelEncoder()
umidade_lbencoder = preprocessing.LabelEncoder()
vento_lbencoder = preprocessing.LabelEncoder()
treinou_lbencoder = preprocessing.LabelEncoder()
```

Naïve Bayes
Exercício - Resposta

Sala Digital

```
#Usando o LabelEncoder para
#atribiur números as variáveis qualitativas
tempo_lbencoder.fit(dados['Tempo'].unique())
umidade_lbencoder.fit(dados['Umidade'].unique())
vento_lbencoder.fit(dados['Vento'].unique())
treinou_lbencoder.fit(dados['Treinou'].unique())
```

```
#Tranformando o dataset de variáveis qualitativas
#para variáveis quantitativas
dados['Tempo'] = tempo_lbencoder.transform(dados['Tempo'])
dados['Umidade'] = umidade_lbencoder.transform(dados['Umidade'])
dados['Vento'] = vento_lbencoder.transform(dados['Vento'])
dados['Treinou'] = treinou_lbencoder.transform(dados['Treinou'])
```

```
#Separando o nosso data set nos atributos previsores
#e na classe objetivo
previsor = dados[['Tempo', 'Umidade', 'Vento']]
classe = dados['Treinou']
```

Naïve Bayes
Exercício - Resposta

Sala Digital

```
#Criando o classificado NaiveBayes
gnb = GaussianNB()
gnb.fit(previsor, classe)
```

```
#Verificando a precisão
print("Precisão =",gnb.score(previsor, classe)*100,"%")
```

```
#Inserindo novos dados para serem previstos
previsao = {'Tempo':['Ensolarado','Nublado', 'Nublado', 'Chuvoso'],
            'Umidade':['Normal', 'Alta', 'Normal', 'Alta'],
            'Vento':['Forte', 'Forte', 'Fraco', 'Forte']}
previsao = pd.DataFrame(data=previsao)

previsao['Tempo'] = tempo_lbencoder.transform(previsao['Tempo'])
previsao['Umidade'] = umidade_lbencoder.transform(previsao['Umidade'])
previsao['Vento'] = vento_lbencoder.transform(previsao['Vento'])
```

Naïve Bayes
Exercícios - Resposta

Sala Digital

```
#Verificando o resultado
print(gnb.predict(previsao))
print(emprestou_lbencoder.inverse_transform(gnb.predict(previsao)))
```

```
[0 1 1 0]
['Não' 'Sim' 'Sim' 'Não']
```

```
#Verificando as probabilidades
gnb.predict_proba(previsao)
```

```
array([[0.6715144 , 0.3284856 ],
       [0.40756566, 0.59243434],
       [0.00771096, 0.99228904],
       [0.86747732, 0.13252268]])
```

Treinamento IOT