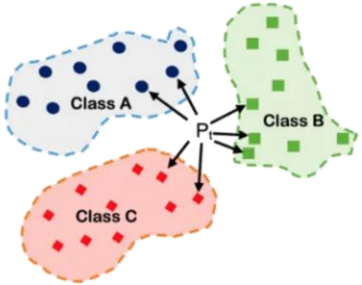
# Universidade Federal de Goiás
# Instituto de Informática
# Prof. Ronaldo Martins da Costa

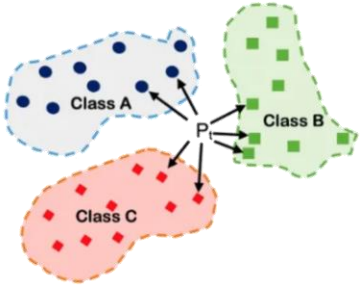



## KNN – K *nearest neighboors* (K Vizinhos mais Próximos)

Supondo que cada característica de seus dados (altura, peso, salário, nota obtida, valor, etc.) seja uma coordenada em um hiperplano

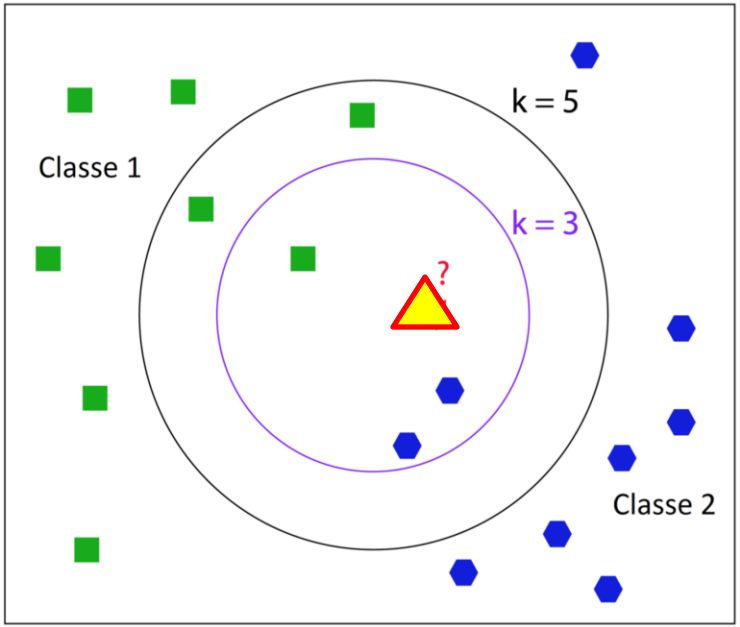
Através de métricas de calculo de distância é possível calcular a distância entre os diversos vetores de características (pontos)

$$D = \sqrt{\left( x_1 - x_2 \right)^2 + \left( y_1 - y_2 \right)^2 + \ldots + \left( z_1 - z_2 \right)^2}$$



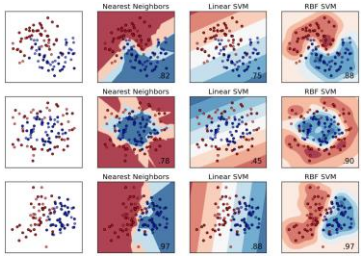

## KNN – K *nearest neighboors* (K Vizinhos mais Próximos)

Sendo assim, é possível calcular a distância entre todos os vetores de características e determinar quais os mais próximos. A acurácia é dada pela equação:




$$AC = \frac{Classificação \quad Correta}{N° \quad Amostras}$$

# “Consumindo” Modelos de IA



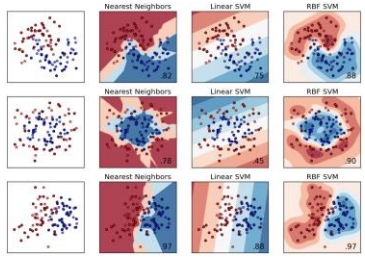

- Os modelos/classificadores podem ser:
  - Discretos

    Atribui cada elemento para uma classe
  - Contínuos

    Atribui uma probabilidade de um determinado elemento pertencer a uma classe, utilizando um limiar de separação entre as classes

- Assim, é necessário que possamos determinar o quão preciso um modelo/classificador retorna resultados

# “Consumindo” Modelos de IA




- Toda classificação atribuirá classes aos elementos mas também está sujeita a erros por diversas razões (qualidade dos dados, falha na técnica de pré-processamento, erros na “limpeza” dos dados, etc.)














# “Consumindo” Modelos de IA

CÃES

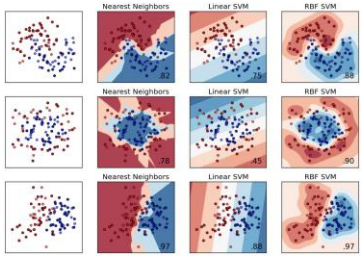
GATOS

# “Consumindo” Modelos de IA

- Vários fatores podem induzir o classificador ao erro



# “Consumindo” Modelos de IA



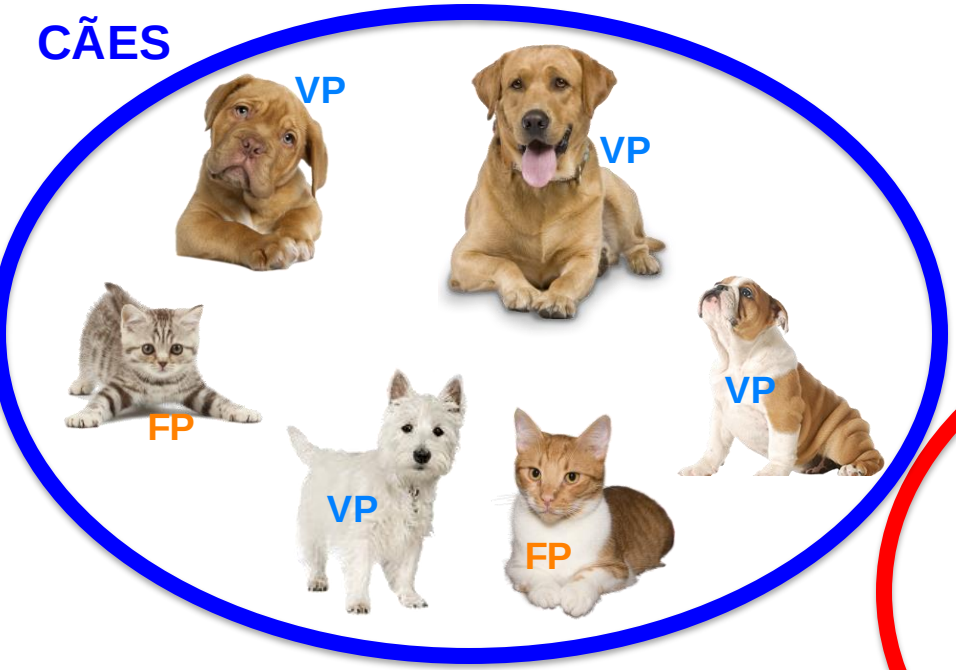

- A sensibilidade do classificador pode ser verificada através da tabela a seguir, muito utilizada em exames clínicos, em aplicações da saúde, mas pode ser aplicada a muitos outros tipos de problemas

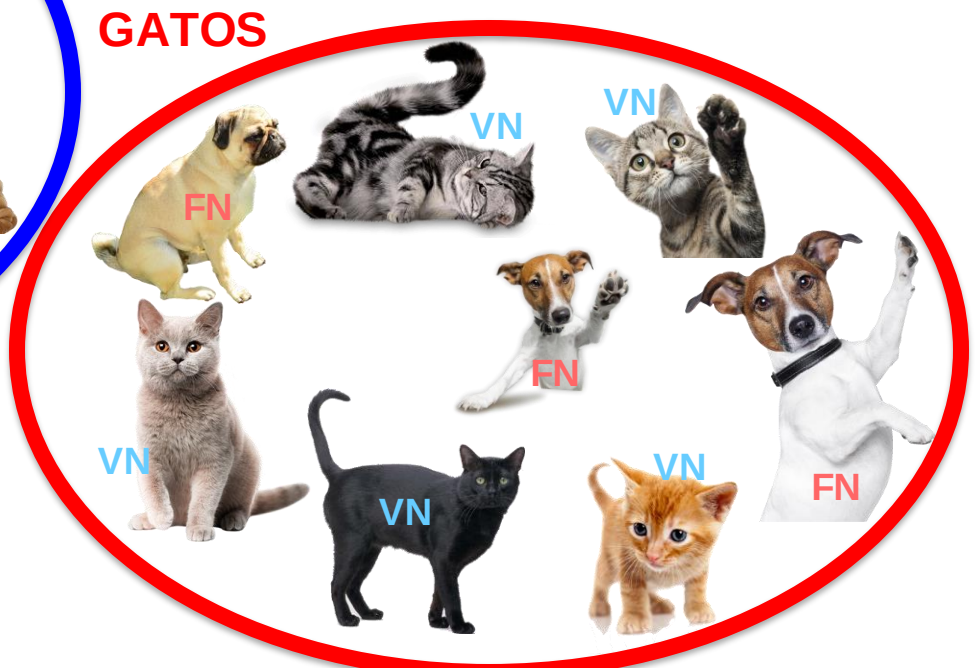
| | | Classe Real | |
|---|---|---|---|
| | | + | − |
| Classe Predita | + | Verdadeiro Positivo (VP) | Falso Positivo (FP) |
| | − | Falso Negativo (FN) | Verdadeiro Negativo (VN) |

Matriz de Confusão para 2 classes

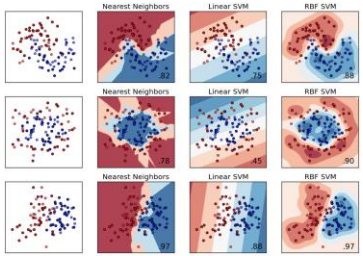
# “Consumindo” Modelos de IA

| | | Classe Real | |
|---|---|---|---|
| | | Cão (+) | Gato (−) |
| Classe Predita | Cão (+) | 4 (VP) | 2 (FP) |
| | Gato (−) | 3 (FN) | 5 (VN) |



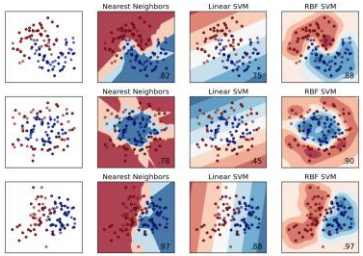



# “Consumindo” Modelos de IA




- Baseado nos resultados da tabela, algumas informações importantes podem ser obtidas como:
  - Sensibilidade
  - Especificidade

| | | Classe Real | | |
|---|---|---|---|---|
| | | Cão (+) | Gato (−) | Total |
| Classe Predita | Cão (+) | 4 (VP) | 2 (FP) | 6 (TP) |
| | Gato (−) | 3 (FN) | 5 (VN) | 8 (TN) |
| | Total | 7 | 7 | 14 |




## Sensibilidade – Equivalente a Taxa de Verdadeiros Positivos (*True Positive Rate* – TPR)

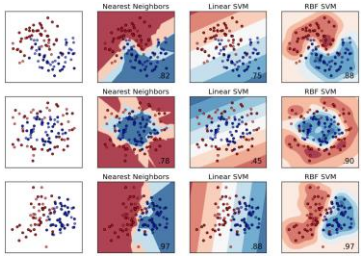
- A sensibilidade de um método reflete o quanto este é eficaz em identificar corretamente, dentre todos os indivíduos avaliados, aqueles que realmente apresentam a característica de interesse

$$S = \frac{VP}{VP + FN}$$

$$S = \frac{4}{4+3}$$

$$S = 0{,}57 \quad ou \quad 57\%$$

| | | Classe Real | | |
|---|---|---|---|---|
| | | Cão (+) | Gato (−) | Total |
| Classe Predita | Cão (+) | 4 (VP) | 2 (FP) | 6 (TP) |
| | Gato (−) | 3 (FN) | 5 (VN) | 8 (TN) |
| | Total | 7 | 7 | 14 |




## Especificidade – Equivalente a Taxa de Falso Positivos (*False Positive Rate* - FPR)

- Reflete o quanto ele é eficaz em identificar corretamente os indivíduos que não apresentam a condição de interesse

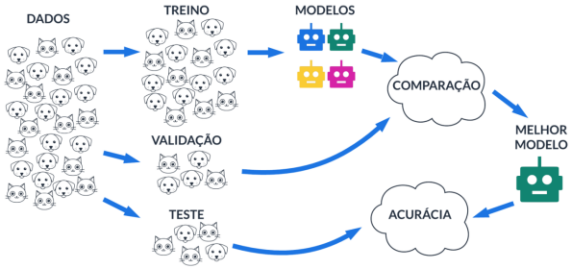
| | | Classe Real | | |
|---|---|---|---|---|
| | | Cão (+) | Gato (−) | Total |
| Classe Predita | Cão (+) | 4 (VP) | 2 (FP) | 6 (TP) |
| | Gato (−) | 3 (FN) | 5 (VN) | 8 (TN) |
| | Total | 7 | 7 | 14 |

$$S = \frac{VN}{VN + FP}$$

$$S = \frac{5}{5+2}$$

$$S = \ 0{,}71 \quad ou \quad 71\%$$

# “Consumindo” Modelos de IA

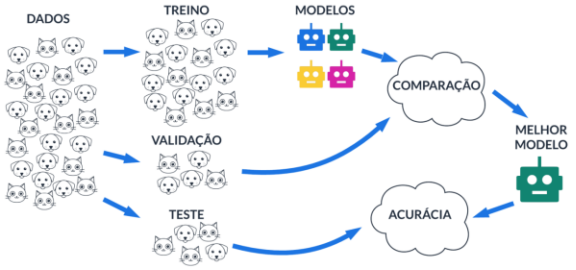



- Para que uma técnica seja estatisticamente robusta, é necessário que alguns aspectos sejam observados

- Tomando como exemplo uma pesquisa eleitoral, para que as "previsões" se aproxime o máximo do resultado real é necessário que os dados representem uma parte significativa da população, possuindo dados de todas as idades, gêneros, classes, regiões geográficas, etc.

- Mas como fazer com que um modelo apresente resultados que representam TODOS os dados de um conjunto ?

# “Consumindo” Modelos de IA

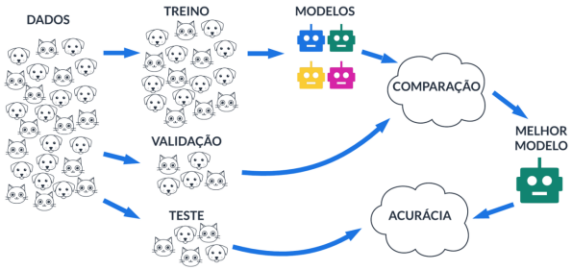



- O objetivo é encontrar um modelo que produza previsões para dados futuros/desconhecidos com erro relativo mínimo

- Dessa forma, “alimentamos” o modelo com uma parte dos dados, a chamada amostra de **treino** (70% a 80%)

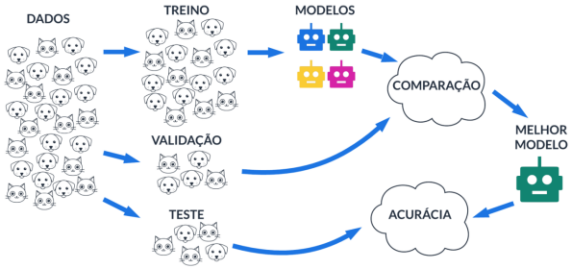
- No entanto, queremos saber sobre o futuro. Para isso utilizamos o modelo para produzir previsões para o restante dos dados, a chamada amostra de **teste** (20% a 30%)

# “Consumindo” Modelos de IA




- A separação dos dados em Treino e Teste, em geral não é suficiente. Uma solução mais adequada é a separação em 3(três) subconjuntos:
- Treino (70% a 80%)
  - Utilizado para treinar o modelo, ou seja, para ajustar os parâmetros internos do modelo (como pesos em redes neurais, coeficientes, etc.)
- Validação (10% a 15%)
  - Durante o treinamento, o desempenho é monitorado no conjunto de validação para ajustar os hiperparâmetros e prevenir o *overfitting*
- Teste (10% a 15%)
  - Após o ajuste e treinamento, o modelo é testado com o conjunto de teste para avaliar sua capacidade de generalizar a dados que nunca foram vistos antes

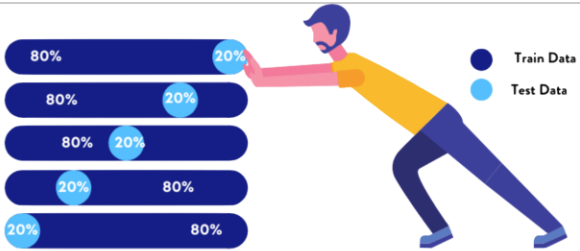
# “Consumindo” Modelos de IA




## *Overfitting*

- Problema que ocorre quando o modelo responde muito bem nos dados de treinamento, mas terá um desempenho ruim em novos dados ou no conjunto de teste, porque o modelo não aprendeu os padrões gerais dos dados, mas sim os ruídos ou as flutuações aleatórias presentes nos dados de treinamento

## Cross Validation




## Validação Cruzada

É uma técnica para avaliar como o resultado de uma análise estatística generalizará para um conjunto independente de dados.

É usado para estimar o quão preciso um classificador será na prática

# “Consumindo” Modelos de IA

## Gerando um classificador

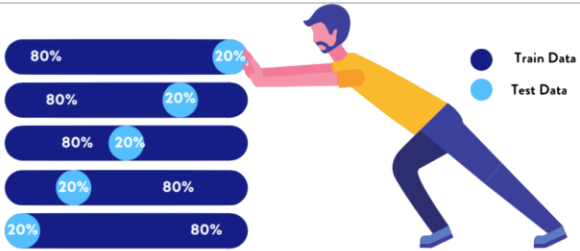

Conjunto de Treino

Conjunto de Teste

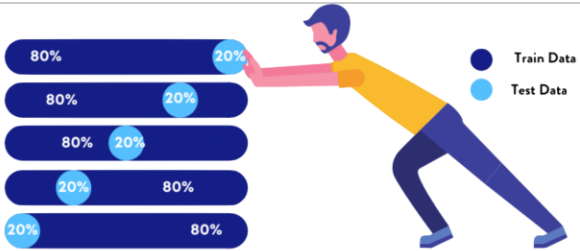
Classificador

| | | Classe Real | |
|---|---|---|---|
| | | + | − |
| Classe Predita | + | (VP) | (FP) |
| | − | (FN) | (VN) |

## Cross Validation



## Cross Validation




## Validação Cruzada


K Fold 1

K Fold 2

K Fold 3

| | K Fold 1 | K Fold 2 + K Fold 3 |
|---|---|---|
| | K Fold 2 | K Fold 1 + K Fold 3 |
| | K Fold 3 | K Fold 1 + K Fold 2 |

Classificador

Classificador

Classificador

| | | Classe Real | |
|---|---|---|---|
| | | + | – |
| Classe Predita | + | (VP) | (FP) |
| | – | (FN) | (VN) |

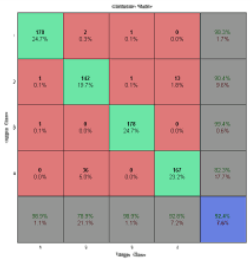
| | | Classe Real | |
|---|---|---|---|
| | | + | – |
| Classe Predita | + | (VP) | (FP) |
| | – | (FN) | (VN) |

| | | Classe Real | |
|---|---|---|---|
| | | + | – |
| Classe Predita | + | (VP) | (FP) |
| | – | (FN) | (VN) |

K = 10

Acurácia = Média

# “Consumindo” Modelos de IA



- Matriz de confusão para N classes

| | | Classe Real | | | |
|---|---|---|---|---|---|
| | | **Cachorro** | **Coelho** | **Gato** | **Lobo** |
| **Classe Predita** | **Cachorro** | **80** | 0 | 2 | 10 |
| | **Coelho** | 10 | **50** | 8 | 5 |
| | **Gato** | 0 | 0 | **70** | 5 |
| | **Lobo** | 10 | 0 | 0 | **100** |

**Total Classificado Corretamente**
**dividido por**
**Total de Elementos**

**300 / 350 = 0,857 ou 85,7**

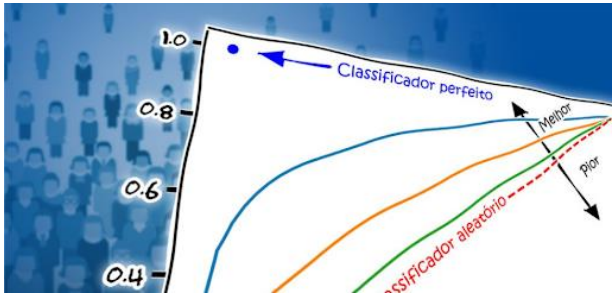
100 Cães

50 Coelhos

80 Gatos

120 Lobos

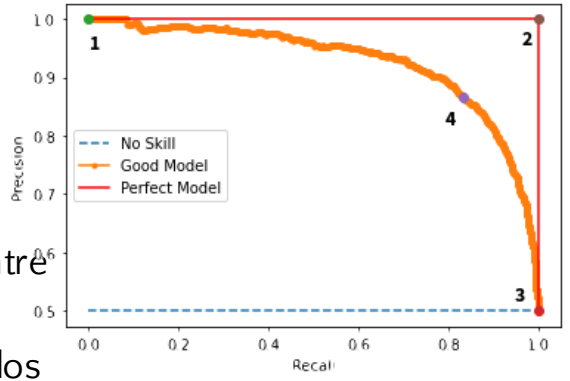
## Curva ROC




- "*Receiver Operating Characteristic*", que pode ser traduzido livremente como “eficiência do operador de recepção de sinais”
- A curva ROC mostra a relação entre a taxa de verdadeiros positivos (TPR) e a taxa de falsos positivos (FPR) para diferentes limiares de decisão
- Eixo X: Taxa de Falso Positivo (*False Positive Rate* - FPR)
- Eixo Y: Taxa de Verdadeiro Positivo (*True Positive Rate* - TPR)
- Ao traçar a curva, o objetivo é observar como a TPR se comporta em função da FPR à medida que o limiar de decisão do modelo é ajustado. O ideal é que o modelo tenha uma TPR alta e uma FPR baixa, o que coloca a curva o mais próximo possível do canto superior esquerdo do gráfico
- O método foi originalmente desenvolvido para avaliar a capacidade de operadores de radar em decidir se uma mancha na tela representava um alvo inimigo ou um aliado

# “Consumindo” Modelos de IA

## Curva *Precision-Recall* (PR)




- Precisão (*Precision*): É a proporção de verdadeiros positivos entre todos os positivos preditos
- Revocação (*Recall*): É a capacidade do modelo de capturar todos os exemplos positivos reais
- Eixo X: *Recall* — A proporção de instâncias positivas corretamente identificadas
- Eixo Y: *Precision* — A proporção de instâncias classificadas como positivas que são realmente positivas
- Assim como a área sob a curva ROC, a área sob a curva *Precision-Recall* é uma métrica de desempenho do modelo. Quanto maior a AUC-PR, melhor o desempenho do modelo, pois ele tem boa precisão e boa revocação em um grande número de diferentes limiares

# “Consumindo” Modelos de IA

**CÃES**

Considerando que o resultado foi:
4 cães e 1 gato

$$\mathrm{Re}\,call = \frac{ClassificaçõesCorretas}{QuantidadeTotalClasse}$$

$$\mathrm{Pr}\,ecision = \frac{ClassificaçõesCorretas}{QuantidadeClassificada}$$

# “Consumindo” Modelos de IA

**CÃES**

Considerando que o resultado foi:
4 cães e 1 gato

$$Recall = \frac{4}{6} = 0,66 \quad ou \quad 66\%$$

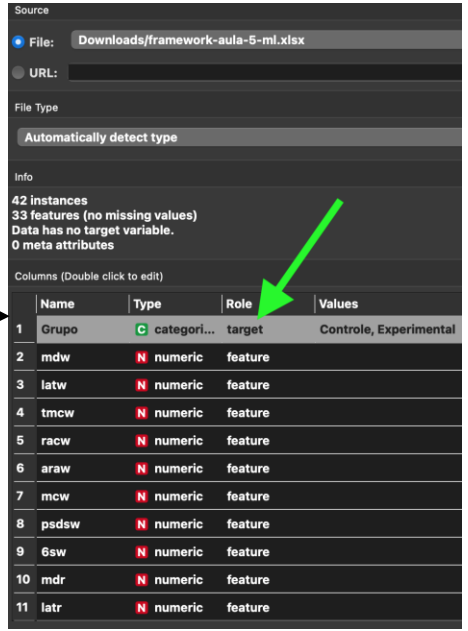
$$Precision = \frac{4}{5} = 0,80 \quad ou \quad 80\%$$



# “Consumindo” Modelos de IA

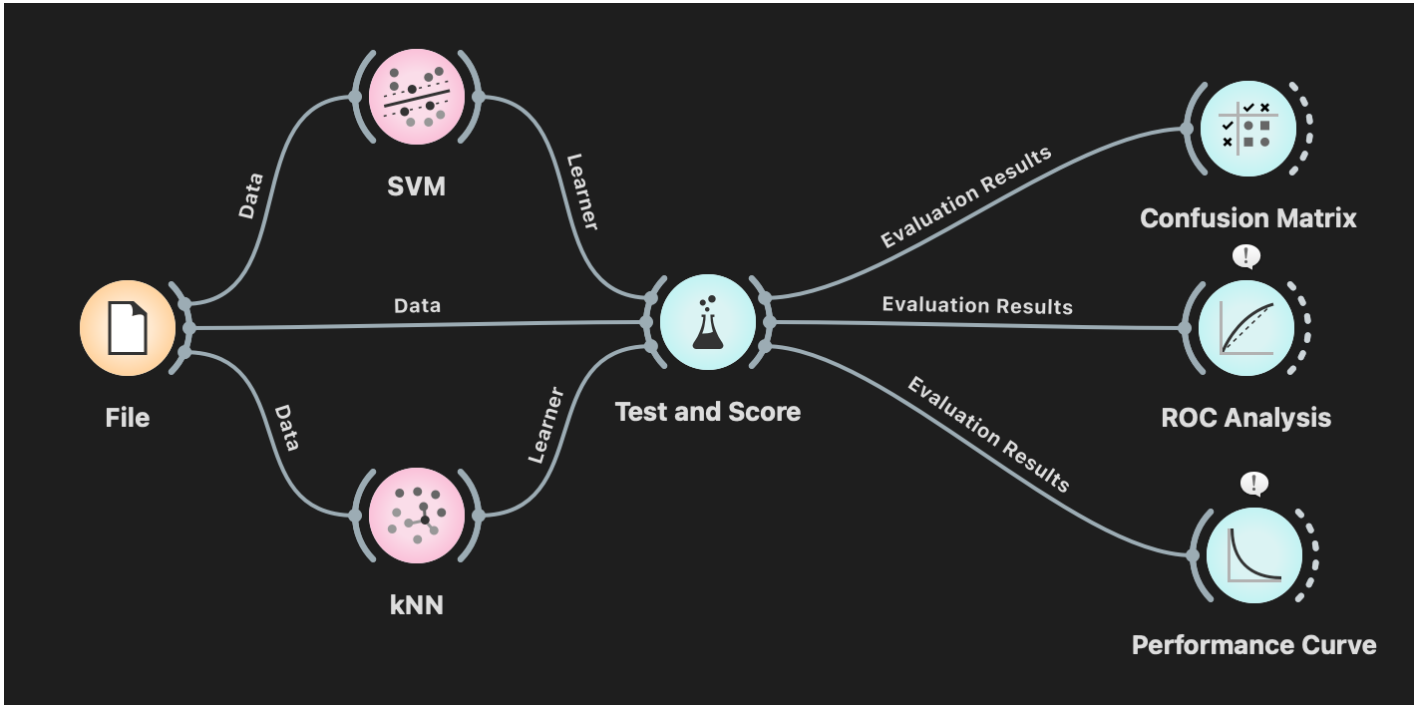
- Realizando um treinamento e validação com o Software Orange
  - Crie um novo projeto, selecione o ícone FILE e associe ao conjunto de dados que você possui, uma das colunas deve ser o TARGET para classificação

New

# “Consumindo” Modelos de IA

- Realizando um treinamento e validação com o Software Orange
  - Selecione os modelos e ferramentas de avaliação desejados



# “Consumindo” Modelos de IA

- Criando um model no DB para receber os valores que serão importados na APP exemplo02

```
from django.db import models

class dados(models.Model):
    grupo = models.CharField(max_length=50, null=True, blank=True,
   verbose_name='Grupo')
    mdw = models.FloatField(null=True, blank=True, verbose_name='mdw')
    latw = models.FloatField(null=True, blank=True, verbose_name='latw')
    tmcw = models.FloatField(null=True, blank=True, verbose_name='tmcw')
    racw = models.FloatField(null=True, blank=True, verbose_name='racw')
    araw = models.FloatField(null=True, blank=True, verbose_name='araw')
    mcw = models.FloatField(null=True, blank=True, verbose_name='mcw')
    psdsw = models.FloatField(null=True, blank=True, verbose_name='psdsw')
    s6w = models.FloatField(null=True, blank=True, verbose_name='s6w')
    mdr = models.FloatField(null=True, blank=True, verbose_name='mdr')
    latr = models.FloatField(null=True, blank=True, verbose_name='latr')
    tmcr = models.FloatField(null=True, blank=True, verbose_name='tmcr')
    racr = models.FloatField(null=True, blank=True, verbose_name='racr')
```

# “Consumindo” Modelos de IA

- Criando um model no DB para receber os valores que serão importados na APP exemplo02

```
arar = models.FloatField(null=True, blank=True, verbose_name='arar')
mcr = models.FloatField(null=True, blank=True, verbose_name='mcr')
psdsr = models.FloatField(null=True, blank=True, verbose_name='psdsr')
s6r = models.FloatField(null=True, blank=True, verbose_name='s6r')
mdg = models.FloatField(null=True, blank=True, verbose_name='mdg')
latg = models.FloatField(null=True, blank=True, verbose_name='latg')
tmcg = models.FloatField(null=True, blank=True, verbose_name='tmcg')
racg = models.FloatField(null=True, blank=True, verbose_name='racg')
arag = models.FloatField(null=True, blank=True, verbose_name='arag')
mcg = models.FloatField(null=True, blank=True, verbose_name='mcg')
psdsg = models.FloatField(null=True, blank=True, verbose_name='psdsg')
s6g = models.FloatField(null=True, blank=True, verbose_name='s6g')
mdwb = models.FloatField(null=True, blank=True, verbose_name='mdwb')
latb = models.FloatField(null=True, blank=True, verbose_name='latb')
tmcb = models.FloatField(null=True, blank=True, verbose_name='tmcb')
racb = models.FloatField(null=True, blank=True, verbose_name='racb')
arab = models.FloatField(null=True, blank=True, verbose_name='arab')
```

# “Consumindo” Modelos de IA

- Criando um model no DB para receber os valores que serão importados na APP exemplo02

```
    mcb = models.FloatField(null=True, blank=True, verbose_name='mcb')
    psdsb = models.FloatField(null=True, blank=True, verbose_name='psdsb')
    s6b = models.FloatField(null=True, blank=True, verbose_name='s6b')

    def __str__(self):
        return self.grupo

    class Meta:
        ordering = ['grupo']
```

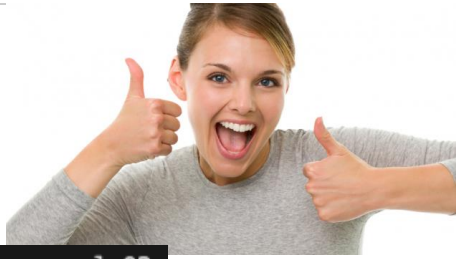
# “Consumindo” Modelos de IA

- Atualizando o modelo no DB

```
python manage.py makemigrations exemplo02

python manage.py migrate
```

# “Consumindo” Modelos de IA



- Se os ajustes no model foram realizados corretamente, você verá uma tela como esta

```
(BigData-env) ronaldocosta@Ronaldos-MacBook-Pro bdpratico % python manage.py makemigrations exemplo02
Migrations for 'exemplo02':
  exemplo02/migrations/0001_initial.py
    - Create model dados
(BigData-env) ronaldocosta@Ronaldos-MacBook-Pro bdpratico % python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, exemplo01, exemplo02, sessions
Running migrations:
  Applying exemplo02.0001_initial... OK
```

# “Consumindo” Modelos de IA


- Na APP Exemplo02, vamos criar as novas rotas

```
from django.contrib import admin
from django.urls import path
from . import views

urlpatterns = [
    path('', views.index, name='index'),
    path('ia_import', views.ia_import, name='ia_import'),
    path('ia_import_save', views.ia_import_save, name='ia_import_save'),
    path('ia_import_list', views.ia_import_list, name='ia_import_list'),
    ]
```

# “Consumindo” Modelos de IA

- Criando as views necessárias na App exemplo02

```
def ia_import(request):
    return render(request, 'ia_import.html')
```

# “Consumindo” Modelos de IA

- Criando as views necessárias na App exemplo02

```
def ia_import_save(request):
    from .models import dados
    import os
    from django.core.files.storage import FileSystemStorage
    if request.method == 'POST' and request.FILES['arq_upload']:
        fss = FileSystemStorage()
        upload = request.FILES['arq_upload']
        file1 = fss.save(upload.name, upload)
        file_url = fss.url(file1)
        from .models import dados
        dados.objects.all().delete()
        i = 0
        file2 = open(file1,'r')
        for row in file2:
            if (i > 0):
                row2 = row.replace(',', '.')
                row3 = row2.split(';')
                dados.objects.create(
                            grupo = row3[0], mdw = float(row3[1]), latw = float(row3[2]),
                            tmcw = float(row3[3]), racw = float(row3[4]), araw = float(row3[5]),
                            mcw = float(row3[6]), psdsw = float(row3[7]), s6w = float(row3[8]),
                            mdr = float(row3[9]), latr = float(row3[10]), tmcr = float(row3[11]),
                            racr = float(row3[12]), arar = float(row3[13]), mcr = float(row3[14]),
```

# “Consumindo” Modelos de IA

- Criando as views necessárias na App exemplo02

```
                        racr = float(row3[12]), arar = float(row3[13]), mcr = float(row3[14]),
                        psdsr = float(row3[15]), s6r = float(row3[16]), mdg = float(row3[17]),
                        latg = float(row3[18]), tmcg = float(row3[19]), racg = float(row3[20]),
                        arag = float(row3[21]), mcg = float(row3[22]), psdsg = float(row3[23]),
                        s6g = float(row3[24]), mdwb = float(row3[25]), latb = float(row3[26]),
                        tmcb = float(row3[27]), racb = float(row3[28]), arab = float(row3[29]),
                        mcb = float(row3[30]), psdsb = float(row3[31]), s6b = float(row3[32]))
            i = i + 1
        file2.close()
        os.remove(file_url.replace("/", ""))
    from django.shortcuts import redirect
    return redirect('ia_import_list')
```

# “Consumindo” Modelos de IA

- Criando as views necessárias na App exemplo02

```
def ia_import_list(request):
    from .models import dados
    data = {}
    data['dados'] = dados.objects.all()
    return render(request, 'ia_import_list.html', data)
```

# “Consumindo” Modelos de IA


- Criando os templates necessárias na App exemplo02

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>IA Import.html</title>
  </head>

  <body>
    <h1>Upload / Import Arquivo!</h1>
    <p>bdpratico\exemplo02\templates\ia_import.html</p>
    <form method="post" enctype="multipart/form-data" action="{% url 'ia_import_save' %}">
      {% csrf_token %}
      <input type="file" name="arq_upload" />
      <button type="submit">Upload</button>
    </form>
  </body>
</html>
```

# “Consumindo” Modelos de IA


- Criando os templates necessárias na App exemplo02

```
{% load static %}
{% load bootstrap5 %}
{% bootstrap_css %}
{% bootstrap_javascript %}
{% bootstrap_messages %}
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>IA Import.html</title>
  </head>

  <body>
    <h1>Arquivo Importado!</h1>
    <p>bdpratico\exemplo02\templates\ia_import_list.html</p>
    <div class="container">
      <div class="row" style="line-height: 1;">
        <div class="col-md-12 col-sm-12 col-xs-12">
          <table class="table table-striped table-hover">
            <th>Grupo</th>
            <th>mdw</th>
            <th>latw</th>
            <th>tmcw</th>
            <th>racw</th>
```

# “Consumindo” Modelos de IA


- Criando os templates necessárias na App exemplo02

```
<th>araw</th>
<th>mcw</th>
<th>psdsw</th>
<th>6sw</th>
<th>mdr</th>
<th>...</th>
<th>mcb</th>
<th>psdsb</th>
<th>6sb</th>
{% for regs in dados %}
  <tr>
    <td>{{ regs.grupo }}</td>
    <td>{{ regs.mdw }}</td>
    <td>{{ regs.latw }}</td>
    <td>{{ regs.tmcw }}</td>
    <td>{{ regs.racw }}</td>
    <td>{{ regs.araw }}</td>
    <td>{{ regs.mcw }}</td>
    <td>{{ regs.psdsw }}</td>
    <td>{{ regs.s6w }}</td>
    <td>{{ regs.mdr }}</td>
    <td>...</td>
    <td>{{ regs.mcb }}</td>
    <td>{{ regs.psdsb }}</td>
```

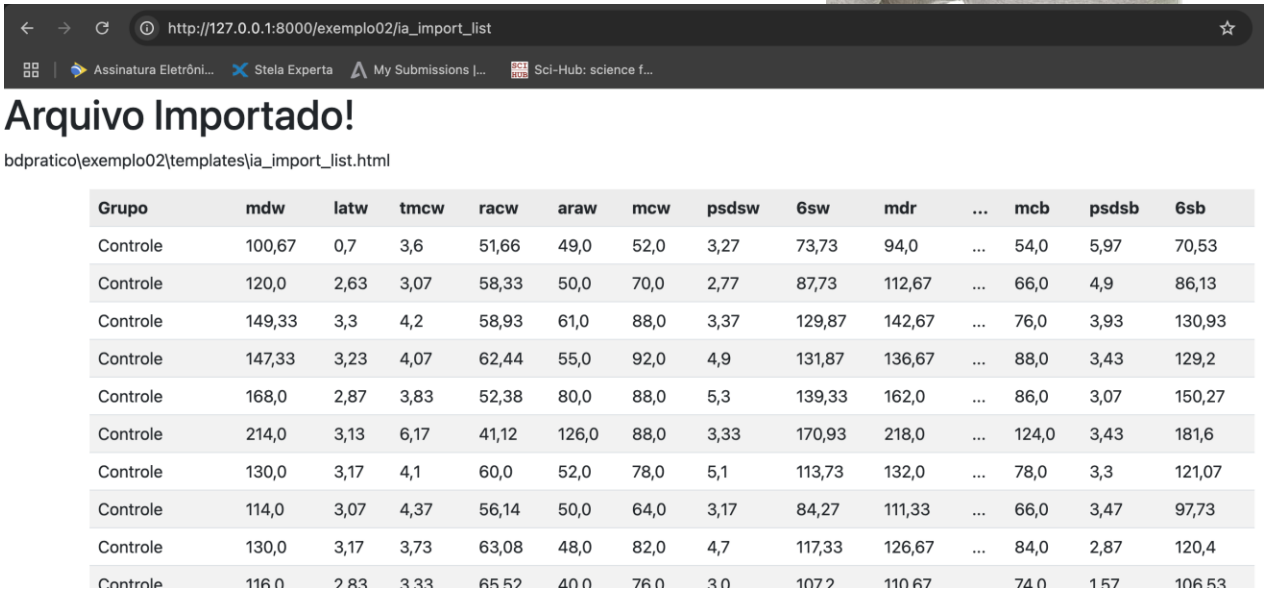
# “Consumindo” Modelos de IA


- Criando os templates necessárias na App exemplo02

```
                    <td>{{ regs.s6b }}</td>
                </tr>
              {% endfor %}
            </table>
          </div>
        </div>
      </div>
    </body>
</html>
```

# “Consumindo” Modelos de IA

- Se os ajustes foram realizados corretamente, após a execução você verá as seguintes telas (exemplo02)


| Grupo | mdw | latw | tmcw | racw | araw | mcw | psdsw | 6sw | mdr | ... | mcb | psdsb | 6sb |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Controle | 100,67 | 0,7 | 3,6 | 51,66 | 49,0 | 52,0 | 3,27 | 73,73 | 94,0 | ... | 54,0 | 5,97 | 70,53 |
| Controle | 120,0 | 2,63 | 3,07 | 58,33 | 50,0 | 70,0 | 2,77 | 87,73 | 112,67 | ... | 66,0 | 4,9 | 86,13 |
| Controle | 149,33 | 3,3 | 4,2 | 58,93 | 61,0 | 88,0 | 3,37 | 129,87 | 142,67 | ... | 76,0 | 3,93 | 130,93 |
| Controle | 147,33 | 3,23 | 4,07 | 62,44 | 55,0 | 92,0 | 4,9 | 131,87 | 136,67 | ... | 88,0 | 3,43 | 129,2 |
| Controle | 168,0 | 2,87 | 3,83 | 52,38 | 80,0 | 88,0 | 5,3 | 139,33 | 162,0 | ... | 86,0 | 3,07 | 150,27 |
| Controle | 214,0 | 3,13 | 6,17 | 41,12 | 126,0 | 88,0 | 3,33 | 170,93 | 218,0 | ... | 124,0 | 3,43 | 181,6 |
| Controle | 130,0 | 3,17 | 4,1 | 60,0 | 52,0 | 78,0 | 5,1 | 113,73 | 132,0 | ... | 78,0 | 3,3 | 121,07 |
| Controle | 114,0 | 3,07 | 4,37 | 56,14 | 50,0 | 64,0 | 3,17 | 84,27 | 111,33 | ... | 66,0 | 3,47 | 97,73 |
| Controle | 130,0 | 3,17 | 3,73 | 63,08 | 48,0 | 82,0 | 4,7 | 117,33 | 126,67 | ... | 84,0 | 2,87 | 120,4 |
| Controle | 116,0 | 2,83 | 3,33 | 65,52 | 40,0 | 76,0 | 3,0 | 107,2 | 110,67 | | 74,0 | 1,57 | 106,53 |

# “Consumindo” Modelos de IA


- Agora vamos transferir os dados para o Pandas e construir o classificador knn

- O primeiro passo é instalar o Pandas e o SkLeanr

```
pip install pandas

e

pip install scikit-learn
```

# “Consumindo” Modelos de IA


- Na APP Exemplo02, vamos criar a nova rota para o treino do KNN

```
from django.contrib import admin
from django.urls import path
from . import views

urlpatterns = [
    path('', views.index, name='index'),
    path('ia_import', views.ia_import, name='ia_import'),
    path('ia_import_save', views.ia_import_save, name='ia_import_save'),
    path('ia_import_list', views.ia_import_list, name='ia_import_list'),
    path('ia_knn_treino', views.ia_knn_treino, name='ia_knn_treino'),
    ]
```

# “Consumindo” Modelos de IA

- Construindo a função que “treina” o KNN na App exemplo02

```
def ia_knn_treino(request):
    data = {}
    print("Vamos ao que interessa...")
    import pandas as pd
    from .models import dados
    dados_queryset = dados.objects.all()
    print("Registros Selecionados.")
    df = pd.DataFrame(list(dados_queryset.values()))
    print("Pandas Carregado e dados 'convertidos'.")
    print("'Cabecalho' dos dados:")
    print(df.head())
    from sklearn.model_selection import train_test_split
    print("Skleanr carregado")

    # Supondo que 'grupo' seja a variável alvo e o restante são as características (features)
    X = df.drop(columns=['grupo', 'id'])  # Variáveis independentes
    y = df['grupo']  # Variável dependente (target)

    # Dividir em treino (70%), teste (15%) e validação (15%)
    X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.30,
    random_state=42)
    X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.50,
    random_state=42)
```

# “Consumindo” Modelos de IA

- Construindo a função que “treina” o KNN na App exemplo02

```
data['dataset'] = X_train.shape
data['treino'] = X_train.shape[0]
data['teste'] = X_test.shape[0]
data['validacao'] = X_val.shape[0]
print(f'Tamanho do conjunto de treino: {X_train.shape}')
print(f'Tamanho do conjunto de teste: {X_test.shape}')
print(f'Tamanho do conjunto de validação: {X_val.shape}')

from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score

# Instanciando o KNN
knn = KNeighborsClassifier()

# Definindo o grid de parâmetros para o KNN
param_grid = {
    'n_neighbors': [3, 5, 7, 9],  # Exemplos de valores possíveis
    'weights': ['uniform', 'distance'],  # Tipos de pesos
    'metric': ['euclidean', 'manhattan']  # Tipos de distância
}
# Usando o GridSearchCV para encontrar os melhores parâmetros
grid_search = GridSearchCV(estimator=knn, param_grid=param_grid, cv=5, verbose=2,
n_jobs=-1)
```

# “Consumindo” Modelos de IA

- Construindo a função que “treina” o KNN na App exemplo02

```
# Treinando o modelo com os dados de treino
grid_search.fit(X_train, y_train)

# Melhor conjunto de parâmetros
data['best'] = grid_search.best_params_
print("Melhores parâmetros encontrados:", grid_search.best_params_)

# Obter o melhor modelo
best_knn = grid_search.best_estimator_

# Previsões no conjunto de validação
y_val_pred = best_knn.predict(X_val)

# Avaliação do modelo (Accuracy)
val_accuracy = accuracy_score(y_val, y_val_pred)
print(f'Acurácia no conjunto de validação: {val_accuracy * 100:.2f}%')
data['acc_validacao'] = round(val_accuracy * 100, 2)

# Previsões no conjunto de teste
y_test_pred = best_knn.predict(X_test)
# Avaliação do modelo no conjunto de teste
test_accuracy = accuracy_score(y_test, y_test_pred)
data['acc_teste'] = round(test_accuracy * 100, 2)
print(f'Acurácia no conjunto de teste: {test_accuracy * 100:.2f}%')
```

# “Consumindo” Modelos de IA

- Construindo a função que “treina” o KNN na App exemplo02

```
import joblib
# Salvar o modelo treinado com o joblib
model_filename = 'knn_model.pkl'  # Caminho do arquivo onde o modelo será salvo
joblib.dump(best_knn, model_filename)
print(f'Modelo salvo em: {model_filename}')
data['file'] = model_filename

return render(request, 'ia_knn_treino.html', data)
```

# “Consumindo” Modelos de IA


- Exibindo os resultados em um template HTML na APP exemplo02

```
{% load static %}
{% load bootstrap5 %}
{% bootstrap_css %}
{% bootstrap_javascript %}
{% bootstrap_messages %}
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>IA Knn.html</title>
  </head>

  <body>
    <h1>Dados do Treinamento - KNN</h1>
    <p>bdpratico\exemplo02\templates\ia_knn_treino.html</p>
    <div class="container">
      <div class="row" style="line-height: 1;">
        <div class="col-md-12 col-sm-12 col-xs-12">
          <table class="table table-striped table-hover">
            <th>Descrição</th>
            <th>Valor</th>
            <tr>
              <td>Dataset:</td>
              <td>{{ dataset }}</td>
```

# “Consumindo” Modelos de IA


- Exibindo os resultados em um template HTML na APP exemplo02

```
</tr>
<tr>
  <td>Treino:</td>
  <td>{{ treino }}</td>
</tr>
<tr>
  <td>Teste:</td>
  <td>{{ teste }}</td>
</tr>
<tr>
  <td>Validação:</td>
  <td>{{ validacao }}</td>
</tr>
<tr>
  <td>Acurácia da Validação:</td>
  <td>{{ acc_validacao }}</td>
</tr>
<tr>
  <td>Acurácia do Teste:</td>
  <td>{{ acc_teste }}</td>
</tr>
```

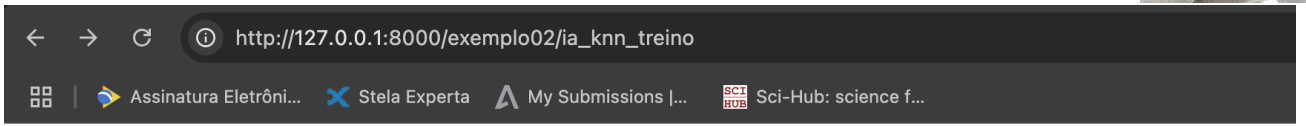
# “Consumindo” Modelos de IA


- Exibindo os resultados em um template HTML na APP exemplo02

```
            <tr>
              <td>Modelo Salvo em:</td>
              <td>{{ file }}</td>
            </tr>
          </table>
        </div>
      </div>
    </div>
  </body>
</html>
```

# “Consumindo” Modelos de IA

- Se os ajustes foram realizados corretamente, após a execução você verá a seguinte tela:




## Dados do Treinamento - KNN

bdpratico\exemplo02\templates\ia_knn_treino.html

| Descrição | Valor |
|---|---|
| Dataset: | (29, 32) |
| Treino: | 29 |
| Teste: | 7 |
| Validação: | 6 |
| Acurácia da Validação: | 83,33 |
| Acurácia do Teste: | 57,14 |
| Modelo Salvo em: | knn_model.pkl |

# “Consumindo” Modelos de IA


- Exibindo a Matriz de confusão de um modelo treinado (App exemplo02):

```
from django.contrib import admin
from django.urls import path
from . import views

urlpatterns = [
    path('', views.index, name='index'),
    path('ia_import', views.ia_import, name='ia_import'),
    path('ia_import_save', views.ia_import_save, name='ia_import_save'),
    path('ia_import_list', views.ia_import_list, name='ia_import_list'),
    path('ia_knn_treino', views.ia_knn_treino, name='ia_knn_treino'),
    path('ia_knn_matriz', views.ia_knn_matriz, name='ia_knn_matriz'),
    ]
```

# “Consumindo” Modelos de IA



- Exibindo a Matriz de confusão de um modelo treinado (App exemplo02):

```
def ia_knn_matriz(request):
    import joblib
    from sklearn.metrics import confusion_matrix
    import numpy as np
    import pandas as pd
    from .models import dados
    dados_queryset = dados.objects.all()
    df = pd.DataFrame(list(dados_queryset.values()))
    from sklearn.model_selection import train_test_split
    X = df.drop(columns=['grupo', 'id'])
    y = df['grupo']
    model_filename = 'knn_model.pkl'
    best_knn = joblib.load(model_filename)
    y_pred = best_knn.predict(X)
    cm = confusion_matrix(y, y_pred)
    data = {
        'matrix': cm.tolist(),
        'labels': np.unique(y).tolist()
    }
    for i in data['matrix']:
        print(i)
    return render(request, 'ia_knn_matriz.html', data)
```

# “Consumindo” Modelos de IA


- Exibindo os resultados em um template HTML – App exemplo02

```
{% load static %}
{% load bootstrap5 %}
{% bootstrap_css %}
{% bootstrap_javascript %}
{% bootstrap_messages %}

<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Matriz de Confusão</title>
  </head>
  <body>
    <div class="container mt-5">
      <h1 class="text-center">Matriz de Confusão</h1>
      <p class="text-center">Exibindo a matriz de confusão gerada pelo modelo</p>

      <table class="table table-bordered table-striped table-hover table-sm">
        <thead class="table-dark">
          <tr>
            <th></th>
```

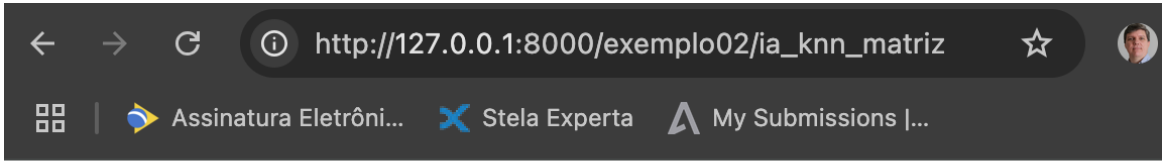
# “Consumindo” Modelos de IA

- Exibindo os resultados em um template HTML – App exemplo02

```
            {% for coluna in labels %}
              <th class="text-center">{{ coluna }}</th>
            {% endfor %}
          </tr>
        </thead>
        <tbody>
          {% for row in matrix %}
            <tr>
              {% if forloop.counter == 1 %}
                <th class="text-end">{{ labels.0 }}</th>
              {% endif %}
              {% if forloop.counter == 2 %}
                <th class="text-end">{{ labels.1 }}</th>
              {% endif %}
              {% for value in row %}
                <td class="text-center">{{ value }}</td>
              {% endfor %}
            </tr>
          {% endfor %}
        </tbody>
      </table>
    </div>
  </body>
</html>
```

# “Consumindo” Modelos de IA

- Se os ajustes foram realizados corretamente, após a execução você verá a seguinte tela:




## Matriz de Confusão

Exibindo a matriz de confusão gerada pelo modelo

| | Controle | Experimental |
|---|---|---|
| **Controle** | 18 | 3 |
| **Experimental** | 7 | 14 |

| | | Classe Real | |
|---|---|---|---|
| | | + | − |
| Classe Predita | + | Verdadeiro Positivo (VP) | Falso Positivo (FP) |
| | − | Falso Negativo (FN) | Verdadeiro Negativo (VN) |

# “Consumindo” Modelos de IA


- Exibindo a Curva ROC de um modelo treinado (App exemplo02):

```
from django.contrib import admin
from django.urls import path
from . import views

urlpatterns = [
    path('', views.index, name='index'),
    path('ia_import', views.ia_import, name='ia_import'),
    path('ia_import_save', views.ia_import_save, name='ia_import_save'),
    path('ia_import_list', views.ia_import_list, name='ia_import_list'),
    path('ia_knn_treino', views.ia_knn_treino, name='ia_knn_treino'),
    path('ia_knn_matriz', views.ia_knn_matriz, name='ia_knn_matriz'),
    path('ia_knn_roc', views.ia_knn_roc, name='ia_knn_roc'),
    ]
```

# “Consumindo” Modelos de IA



- Exibindo a Curva ROC de um modelo treinado (App exemplo02):

```
def ia_knn_roc(request):
    import joblib
    import pandas as pd
    from sklearn.metrics import roc_curve, auc
    import plotly.graph_objects as go
    import numpy as np
    from .models import dados
    from django.shortcuts import render

    dados_queryset = dados.objects.all()
    df = pd.DataFrame(list(dados_queryset.values()))
    X = df.drop(columns=['grupo', 'id'])
    y = df['grupo'].map({'Controle': -1, 'Experimental': 1})
    model_filename = 'knn_model.pkl'
    best_knn = joblib.load(model_filename)
    y_pred_prob = best_knn.predict_proba(X)[:, 1]
    fpr, tpr, thresholds = roc_curve(y, y_pred_prob)
    roc_auc = auc(fpr, tpr)
```

# “Consumindo” Modelos de IA



- Exibindo a Curva ROC de um modelo treinado (App exemplo02):

```
    fig = go.Figure()
    fig.add_trace(go.Scatter(x=fpr, y=tpr, mode='lines', name=f'ROC Curve (AUC =
    {roc_auc:.2f})', line=dict(color='blue')))
    fig.add_trace(go.Scatter(x=[0, 1], y=[0, 1], mode='lines', name='Random Classifier',
    line=dict(dash='dash', color='gray')))
    fig.update_layout(
        title='Curva ROC',
        xaxis_title='Taxa de Falsos Positivos (FPR)',
        yaxis_title='Taxa de Verdadeiros Positivos (TPR)',
        showlegend=True
    )
    graph = fig.to_html(full_html=False)
    return render(request, 'ia_knn_roc.html', {'graph': graph})
```

# “Consumindo” Modelos de IA

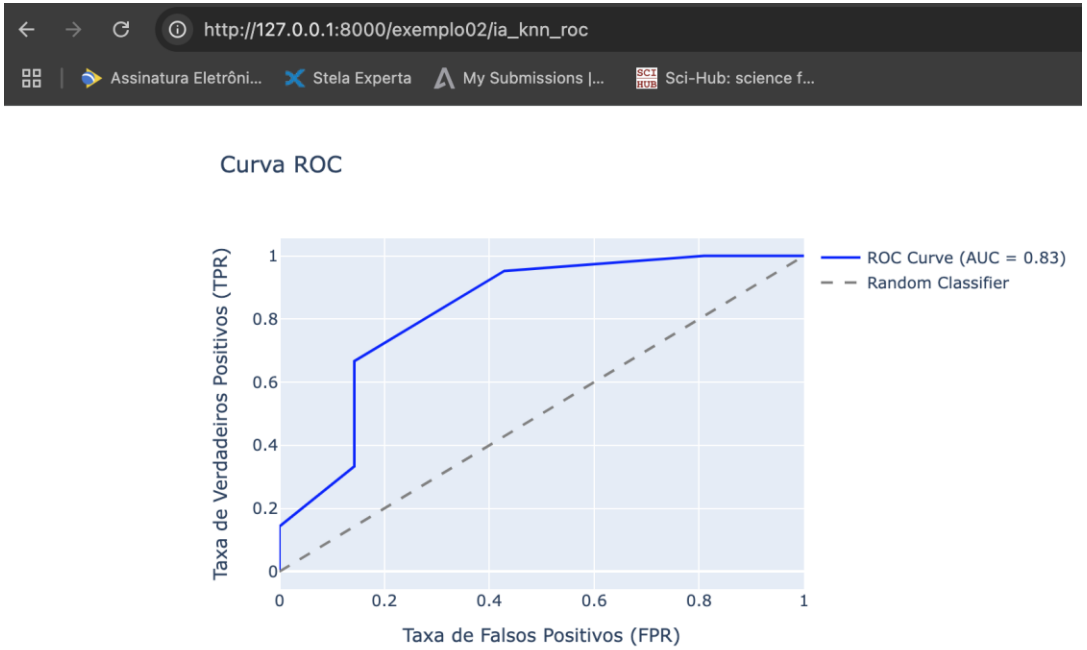

- Exibindo os resultados em um template HTML – App exemplo02

```
{% load static %}
{% load bootstrap5 %}
{% bootstrap_css %}
{% bootstrap_javascript %}
{% bootstrap_messages %}

<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Curva ROC</title>
  </head>
  <body>
    <div class="container mt-5">
      <h1 class="text-center">Curva ROC do Modelo KNN</h1>
      <div class="text-center">
        {{ graph|safe }}
      </div>
    </div>
  </body>
</html>
```

# “Consumindo” Modelos de IA

- Se os ajustes foram realizados corretamente, após a execução você verá a seguinte tela:



# “Consumindo” Modelos de IA


- Exibindo a Curva Recall de um modelo treinado (App exemplo02):

```
from django.contrib import admin
from django.urls import path
from . import views

urlpatterns = [
    path('', views.index, name='index'),
    path('ia_import', views.ia_import, name='ia_import'),
    path('ia_import_save', views.ia_import_save, name='ia_import_save'),
    path('ia_import_list', views.ia_import_list, name='ia_import_list'),
    path('ia_knn_treino', views.ia_knn_treino, name='ia_knn_treino'),
    path('ia_knn_matriz', views.ia_knn_matriz, name='ia_knn_matriz'),
    path('ia_knn_roc', views.ia_knn_roc, name='ia_knn_roc'),
    path('ia_knn_recall', views.ia_knn_recall, name='ia_knn_recall'),
    ]
```

# “Consumindo” Modelos de IA

- Exibindo a Curva Recall de um modelo treinado (App exemplo02):

```
def ia_knn_recall(request):
    import joblib
    import pandas as pd
    from sklearn.metrics import precision_recall_curve, auc
    import plotly.graph_objects as go
    import numpy as np
    from .models import dados
    from django.shortcuts import render

    dados_queryset = dados.objects.all()
    df = pd.DataFrame(list(dados_queryset.values()))
    X = df.drop(columns=['grupo', 'id'])
    y = df['grupo'].map({'Controle': -1, 'Experimental': 1})
    model_filename = 'knn_model.pkl'
    best_knn = joblib.load(model_filename)
    y_pred_prob = best_knn.predict_proba(X)[:, 1]
    precision, recall, thresholds = precision_recall_curve(y, y_pred_prob)
    pr_auc = auc(recall, precision)
```

# “Consumindo” Modelos de IA

- Exibindo a Curva Recall de um modelo treinado (App exemplo02):

```
    fig = go.Figure()
    fig.add_trace(go.Scatter(x=recall, y=precision, mode='lines', name=f'Precision-Recall
    Curve (AUC = {pr_auc:.2f})', line=dict(color='blue')))
    fig.add_trace(go.Scatter(x=[0, 1], y=[0, 0], mode='lines', name='Random Classifier',
    line=dict(dash='dash', color='gray')))
    fig.update_layout(
        title='Curva Precision-Recall',
        xaxis_title='Recall',
        yaxis_title='Precision',
        showlegend=True
    )
    graph = fig.to_html(full_html=False)
    return render(request, 'ia_knn_recall.html', {'graph': graph})
```

# “Consumindo” Modelos de IA


- Exibindo os resultados em um template HTML – App exemplo02

```
{% load static %}
{% load bootstrap5 %}
{% bootstrap_css %}
{% bootstrap_javascript %}
{% bootstrap_messages %}

<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Curva Precision-Recall</title>
  </head>
  <body>
    <div class="container mt-5">
      <h1 class="text-center">Curva Precision-Recall do Modelo KNN</h1>
      <div class="text-center">
        {{ graph|safe }}
      </div>
    </div>
  </body>
</html>
```

# “Consumindo” Modelos de IA

- Se os ajustes foram realizados corretamente, após a execução você verá a seguinte tela:

# Universidade Federal de Goiás
# Instituto de Informática
# Prof. Ronaldo Martins da Costa