

UFSC - Campus Araranguá  
Curso de Engenharia da Computação  
Disciplina de Tópicos Especiais em Ciência de Dados

# Classificação e Clusterização com dados do Enem por Escola

Gabriel Estevam de Oliveira  
Luan Lorenzo dos Santos Borges  
Roger Eliodoro Condras



# Introdução

- Conjunto de dados **Enem por Escola** disponibilizado pelo INEP (Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira)
- Dados da prova do Enem entre 2005 e 2015.

# Objetivos

- Aplicar técnicas de limpeza, transformação e visualização dos dados.
- Aplicar uma técnica de clusterização a fim de determinar as classes.
- Aplicar diferentes algoritmos de classificação.
- Extrair informações a partir dos resultados obtidos.

# Conjunto de Dados

- Manual de utilização dos dados.
- Dicionário de dados.
- Arquivos SAS e SPSS.
- Arquivo CSV.

# Conjunto de Dados

- 172305 tuplas com 27 colunas.
- Atributos:

Variável	Descrição	Edição que contém a Informação
NU_ANO	Ano da edição do Enem	2005 a 2015
CO_UF_ESCOLA	Código do estado	2005 a 2015
SG_UF_ESCOLA	Sigla do estado	2005 a 2015
CO_MUNICIPIO_ESCOLA	Código do município	2005 a 2015
NO_MUNICIPIO_ESCOLA	Nome do município	2005 a 2015
CO_ESCOLA_EDUCACENSO	Código da escola no Educacenso	2005 a 2015

# Conjunto de Dados

Variável	Descrição	Edição que contém a Informação
NO_ESCOLA_EDUCACENSO	Nome da escola	2005 a 2015
TP_DEPENDENCIA ADM_ESCOLA	1 - Federal, 2 - Estadual, 3 - Municipal, 4 - Privada.	2005 a 2015
TP_LOCALIZACAO_ESCOLA	1 - Urbana, 2 - Rural	2005 a 2015
NU_MATRICULAS	Número de alunos matriculados	2005 a 2015
NU_PARTICIPANTES_NEC_ESP	Número de participantes com necessidades especiais.	2013 a 2015
NU_PARTICIPANTES	Número de alunos participantes	2005 a 2015
NU_TAXA_PARTICIPACAO	Taxa de participação	2009 a 2015
NU_MEDIA_CN	Média das notas de Ciências da Natureza.	2009 a 2015

# Conjunto de Dados

Variável	Descrição	Edição que contém a Informação
NU_MEDIA_CH	Média das notas de Ciências Humanas.	2009 a 2015
NU_MEDIA_LP	Média das notas de Linguagens e Códigos.	2009 a 2015
NU_MEDIA_MT	Média das notas de Matemática.	2009 a 2015
NU_MEDIA_RED	Média das notas de Redação.	2008 a 2015
NU_MEDIA_OBJ	Média das notas da prova objetiva.	2008
NU_MEDIA_TOT	Média total (RED + OBJ).	2005 a 2007
INSE	Indicador socioeconômico.	2015

# Conjunto de Dados

Variável	Descrição	Edição que contém a Informação
PC_FORMACAO_DOCENTE	Adequação da formação do docente.	2013 a 2015
NU_TAXA_PERMANENCIA	Indicador de permanência na Escola	2014 a 2015
NU_TAXA_APROVACAO	Taxa de aprovação no ensino médio.	2005, 2007 a 2015
NU_TAXA_REPROVACAO	Taxa de reprovação no ensino médio	2005, 2007 a 2015
NU_TAXA_ABANDONO	Taxa de abandono no ensino médio	2005, 2007 a 2015
PORTE_ESCOLA	1 a 30 alunos, 31 a 60 alunos 61 a 90 alunos, Maior que 90 alunos	2005 a 2015

# Limpeza e Transformações

- Corte dos dados com ano < 2009:

```
ds = ds[ds['NU_ANO'] >= 2009]
```

- Corte de colunas com valores nulos, redundantes ou sem valor agregado:

```
ds.drop(['CO_UF_ESCOLA', 'CO_MUNICIPIO_ESCOLA', 'CO_ESCOLA_EDUCACENSO',
         'NU_PARTICIPANTES_NEC_ESP', 'NU_MEDIA_OBJ', 'NU_MEDIA_TOT',
         'NU_TAXA_PERMANENCIA', 'NU_TAXA_REPROVACAO', 'NU_TAXA_ABANDONO'],
         axis=1, inplace=True)
```

# Limpeza e Transformações

- Substituição de valores nulos na nota da redação:

```
ds['NU_MEDIA_RED'].fillna(0, inplace=True)
```

- Criação do atributo média total:

```
ds['MEDIA_TOT'] = (ds['NU_MEDIA_CN'] + ds['NU_MEDIA_CH']  
+ ds['NU_MEDIA_LP'] + ds['NU_MEDIA_MT']  
+ ds['NU_MEDIA_RED'])/5
```

# Limpeza e Transformações

## - Discretização dos Dados:

```
# Regioes: Norte(1), Nordeste(2), Centro-Oeste(3), Sudeste(4), Sul(5)
dic = {'AM':1,'RR':1,'AP':1,'PA':1,'TO':1,'RO':1,'AC':1,
       'MA':2,'PI':2,'CE':2,'RN':2,'PE':2,'PB':2,'SE':2,'AL':2,'BA':2,
       'MT':3,'MS':3,'GO':3,'DF':3,
       'SP':4,'RJ':4,'ES':4,'MG':4,
       'PR':5,'RS':5,'SC':5
    }
def discretizacao(x):
    return dic[x]
ds['REGIAO'] = ds['SG_UF_ESCOLA'].apply(discretizacao)

classes = [0,0,0,0]
for i in ds['MEDIA_TOT_DIS']:
    classes[i-1] += 1
for i in classes:
    print(i/sum(classes))
```

```
0.2494483555742356
0.25155941043300506
0.24917133932579977
0.24982089466695961
```

# Limpeza e Transformações

## - Discretização dos Dados:

```
dic = {'De 1 a 30 alunos':1, 'De 31 a 60 alunos':2,
       'De 61 a 90 alunos':3, 'Maior que 90 alunos':4}
def discretizacao(x):
    return dic[x]
ds['PORTE_ESCOLA_DIS'] = ds['PORTE_ESCOLA'].apply(discretizacao)
```

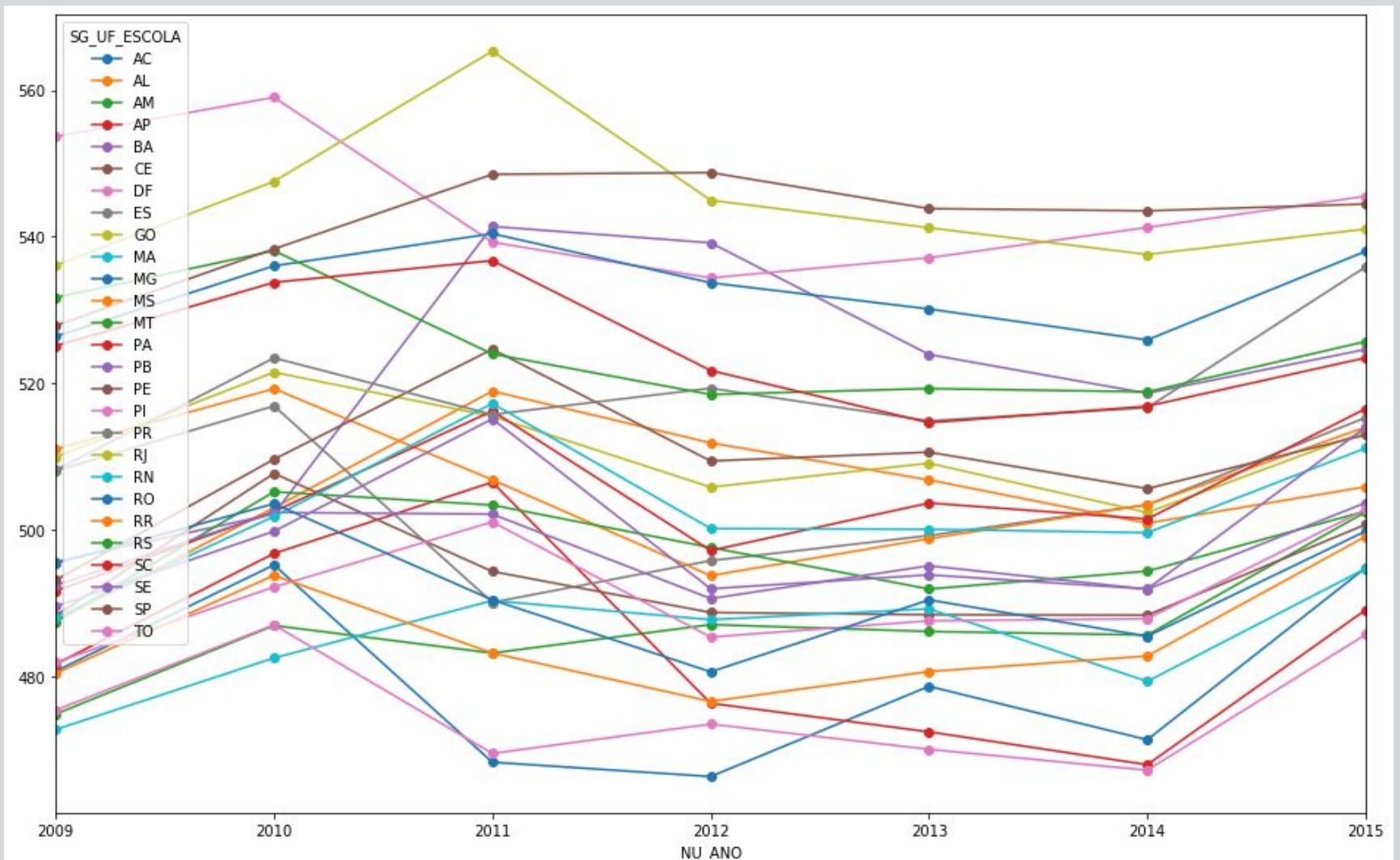
# Limpeza e Transformações

- Discretização dos Dados:

```
def discretizacao(x):
    if x < 480:
        return 1
    elif x < 509:
        return 2
    elif x < 556:
        return 3
    else:
        return 4
ds['MEDIA_TOT_DIS'] = ds['MEDIA_TOT'].apply(discretizacao)
```

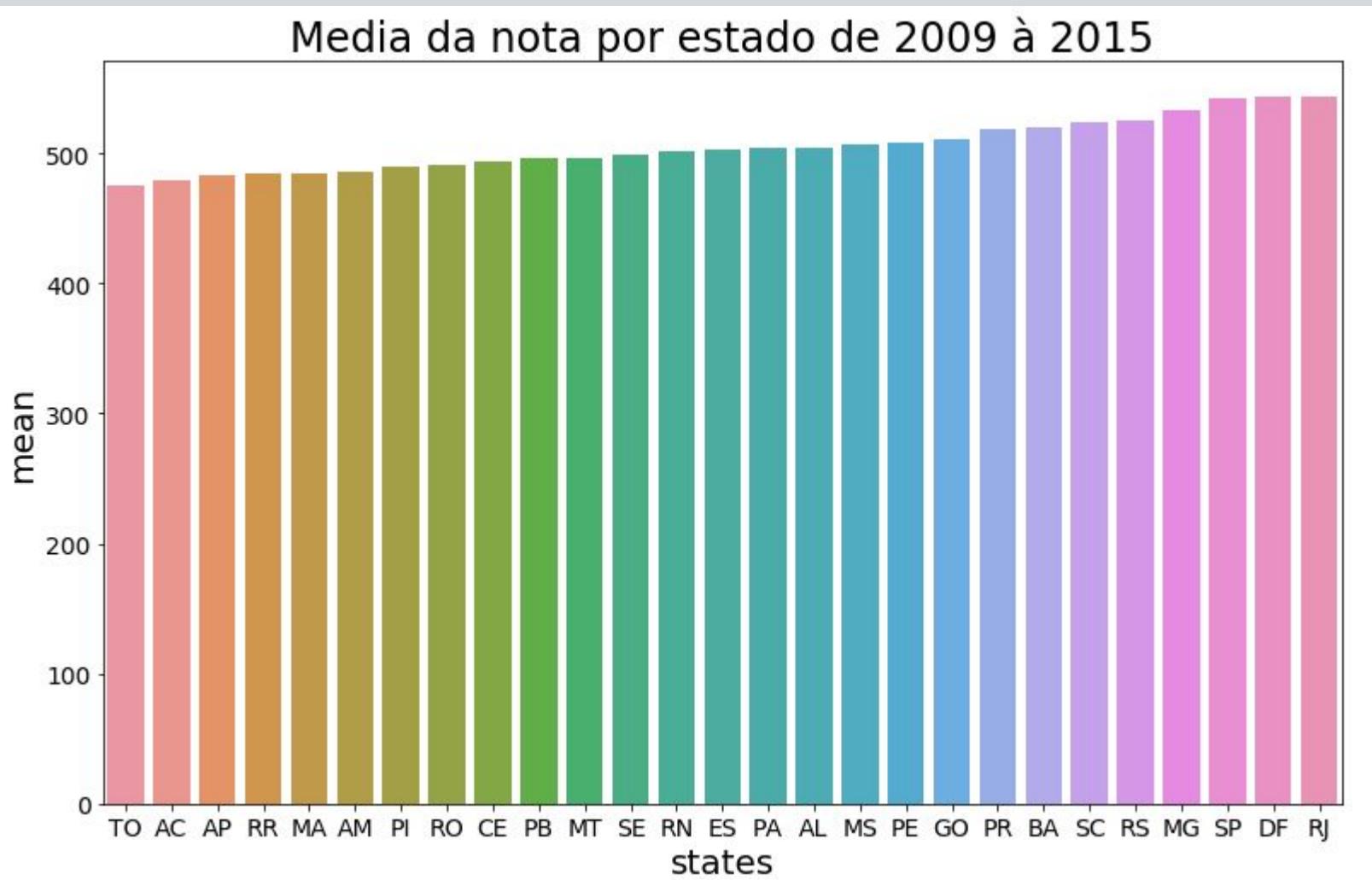
# Visualização

## - Média anual por Estado



# Visualização

## - Média total por Estado



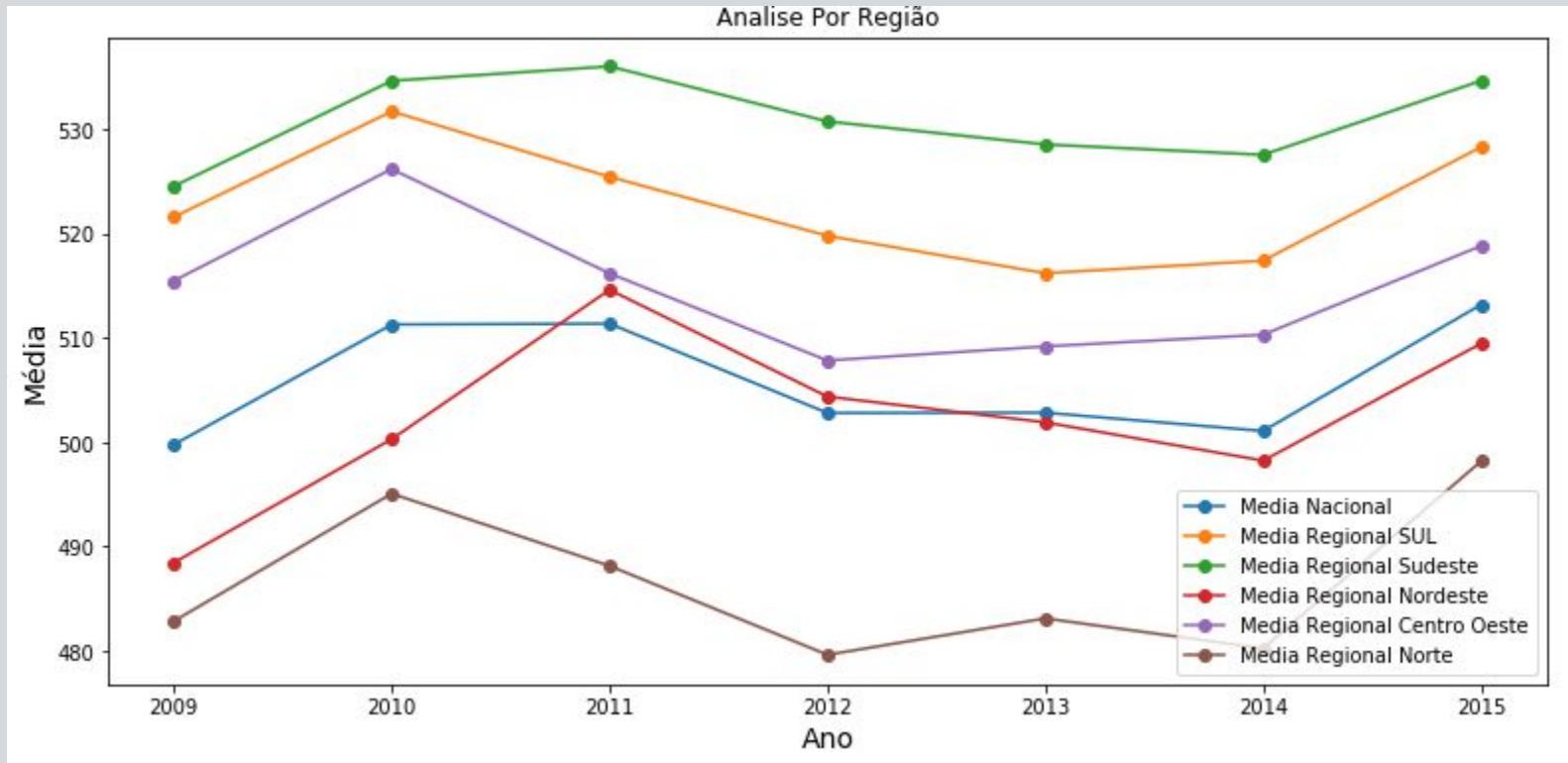
# Visualização

## - Média Nacional por ano



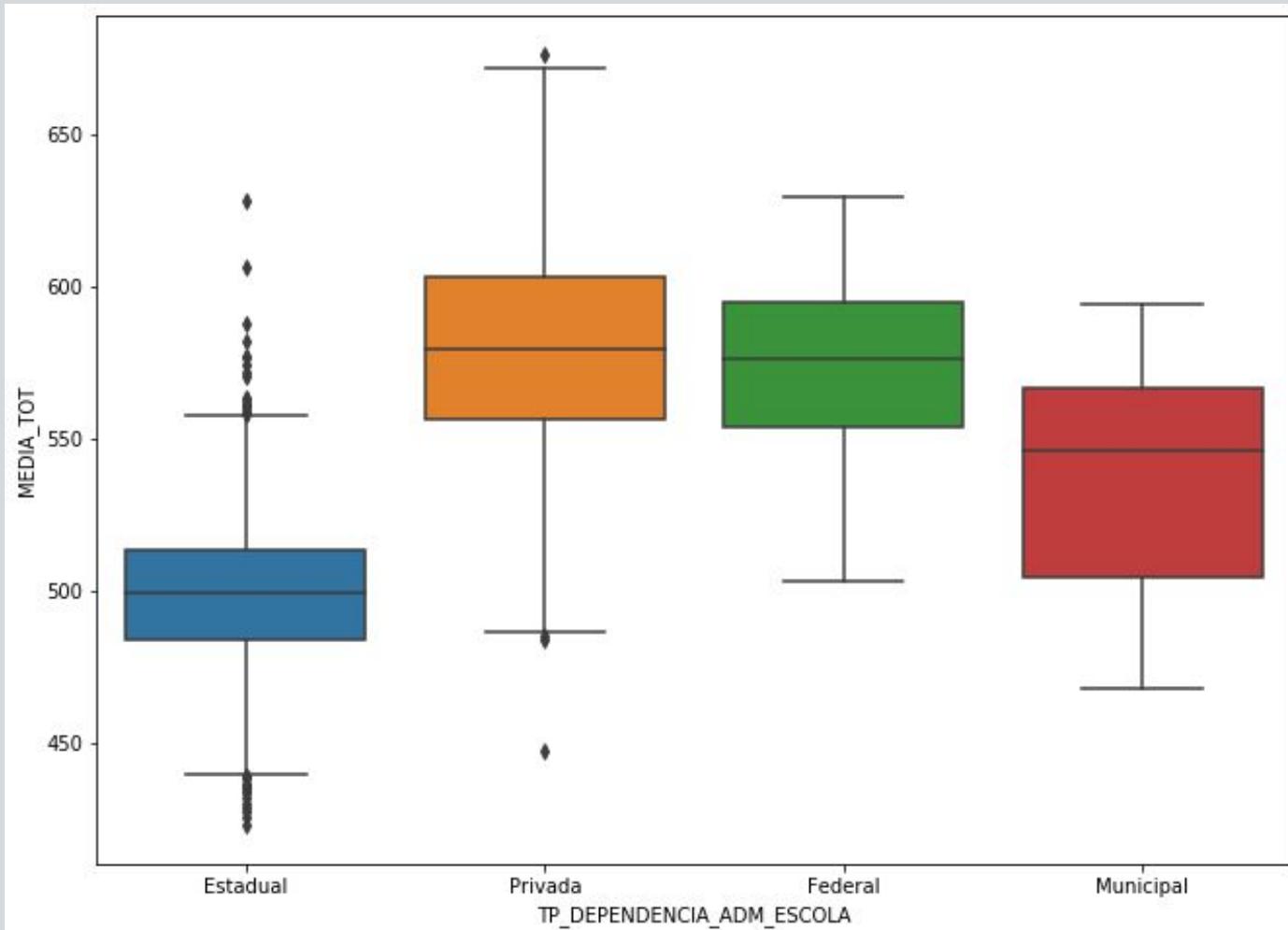
# Visualização

## - Média anual por região



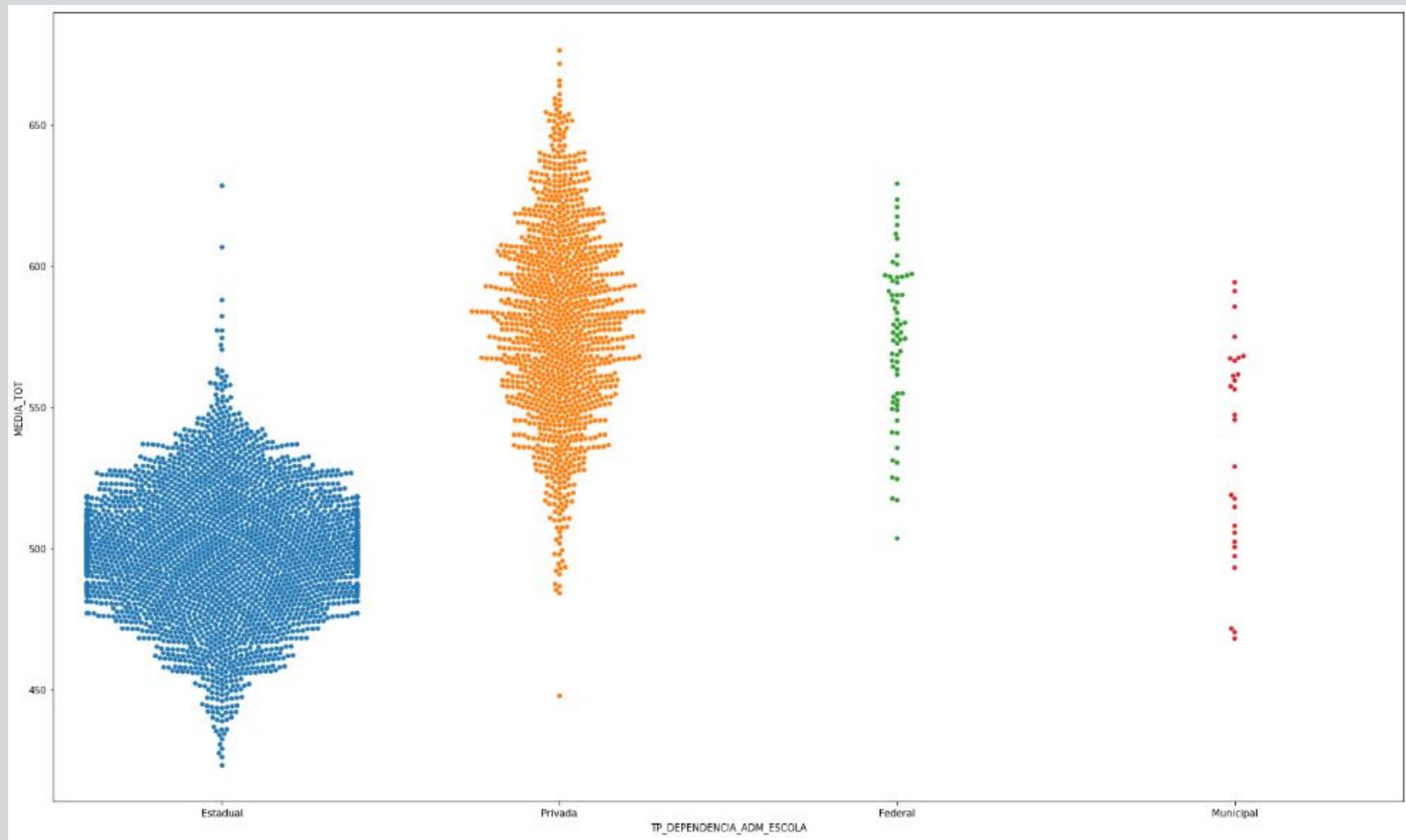
# Visualização - Santa Catarina

- Plotagem de caixa considerando média total e o tipo de dependência administrativa da escola



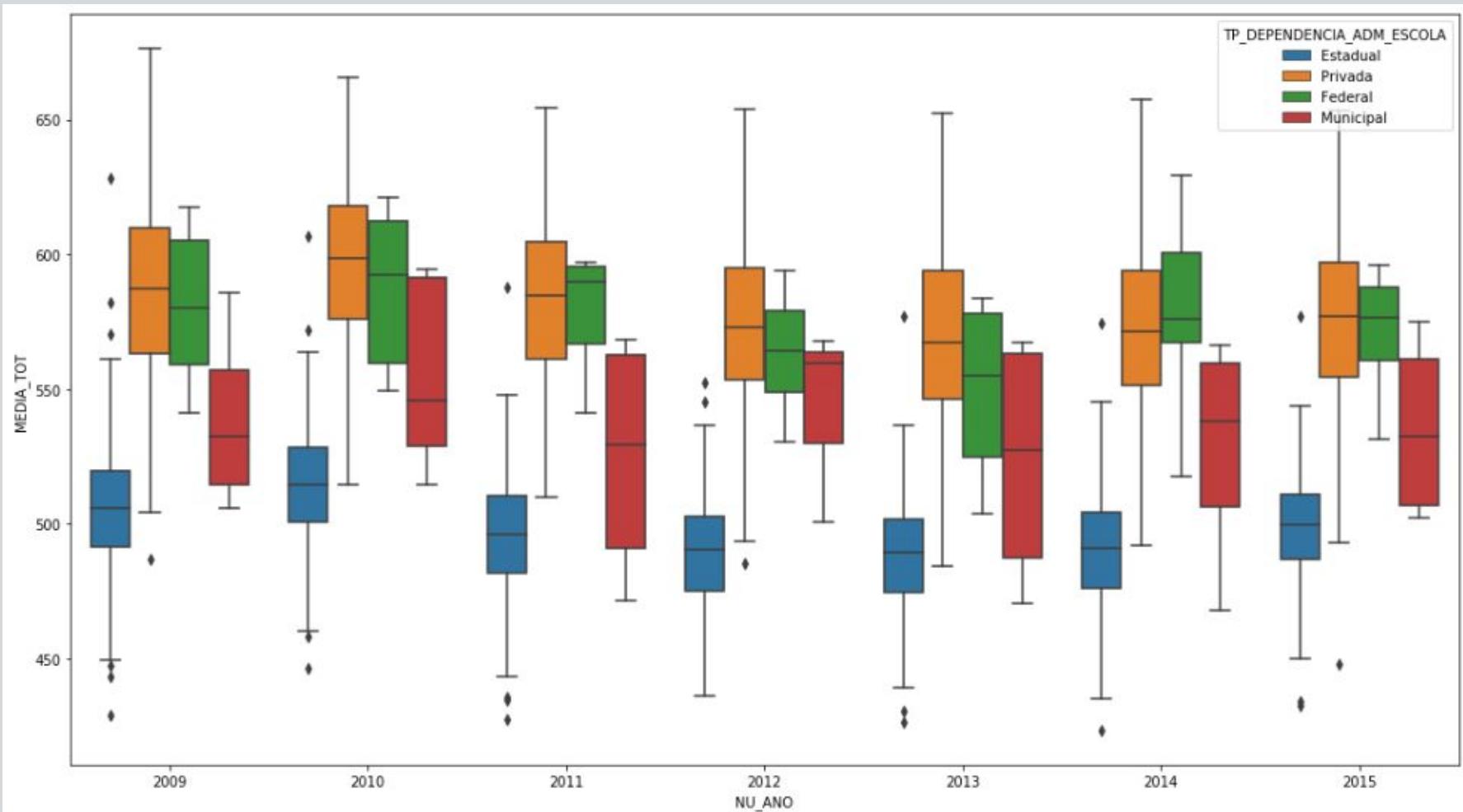
# Visualização - Santa Catarina

- Plotagem de enxame considerando média total e o tipo de dependência administrativa da escola



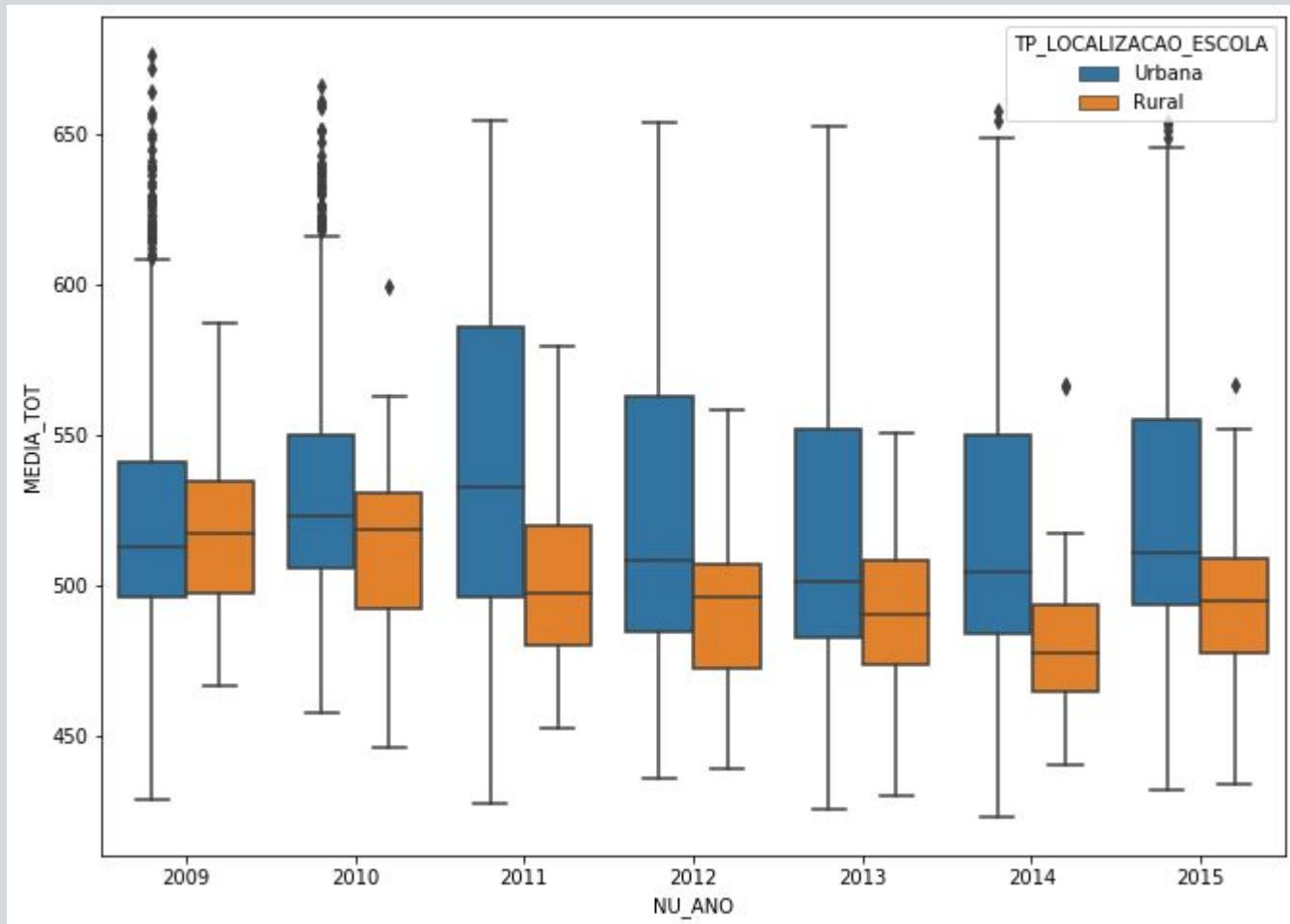
# Visualização - Santa Catarina

- Plotagem de caixa em cada ano considerando a média total de cada escola e seu tipo de dependência administrativa



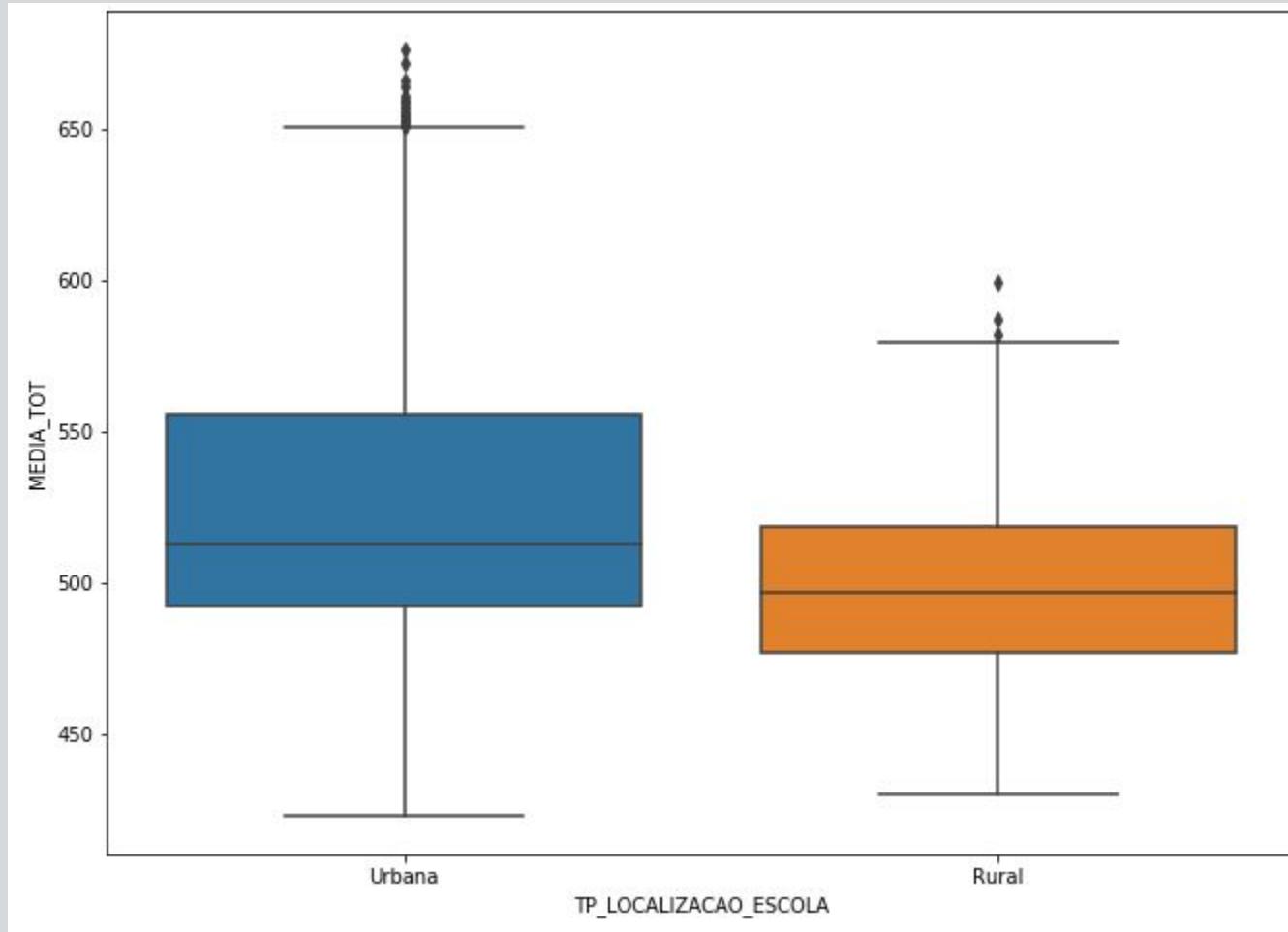
# Visualização - Santa Catarina

- Plotagem de caixa considerando a média total e o tipo de localização da escola em cada ano



# Visualização - Santa Catarina

- Plotagem de caixa considerando a média total e o tipo de localização da escola



# Classificação

## - K vizinhos:

```
from sklearn.neighbors import KNeighborsClassifier  
  
model = KNeighborsClassifier(n_neighbors=4)  
  
# Treina o modelo utilizando o conjunto de treinamento  
model.fit(X_train,y_train)  
  
# Prediz a resposta para o conjunto de dados de teste  
y_pred = model.predict(X_test)  
  
# Apresenta a acurácia do modelo  
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))  
  
Accuracy: 0.5056834463654599
```

# Classificação

## - Naive Bayes:

```
# Importa o classificador
from sklearn.naive_bayes import GaussianNB

# Cria o classificador
gnb = GaussianNB()

# Treina o modelo usando o conjunto de treinamento
gnb.fit(X_train, y_train)

# Prediz a resposta para o conjunto de teste
y_pred = gnb.predict(X_test)

# Acurácia do Modelo; com que frequência o classificador está correto?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

Accuracy: 0.5479351736873945

# Classificação

## - Random Forest:

```
from sklearn.ensemble import RandomForestClassifier

# Cria o classificador
clf=RandomForestClassifier(n_estimators=100)

# Treina o modelo usando os conjuntos de treinamento (características e rótulo para p
clf.fit(X_train,y_train)

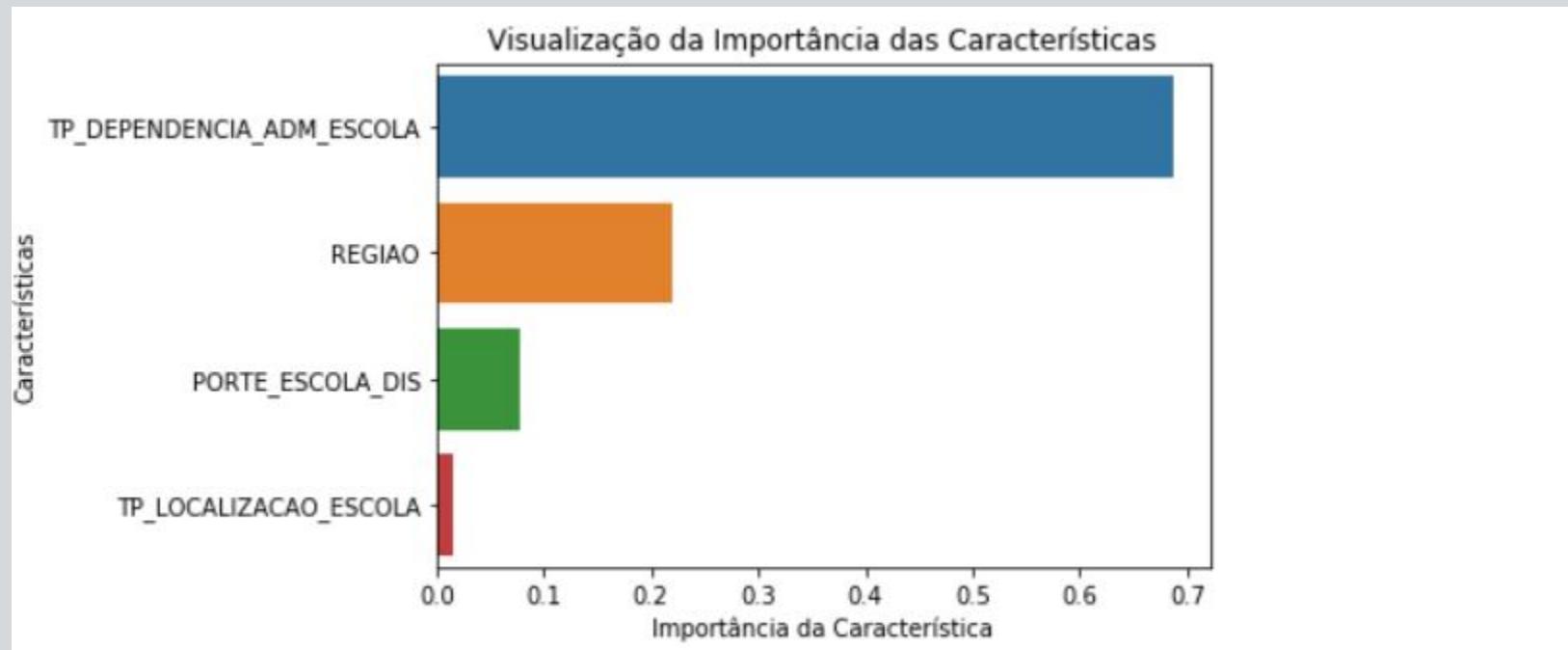
# Realiza a predição utilizando o conjunto de teste
y_pred=clf.predict(X_test)

# Apresenta a acurácia do modelo
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

Accuracy: 0.5782150476008533

# Classificação

- Importância das Características:



# Classificação

## - Árvore de decisão:

```
# Cria o objeto classificador para a árvore de decisão
clf = DecisionTreeClassifier(max_depth=4)

# Treina o objeto classificador
clf = clf.fit(X_train,y_train)

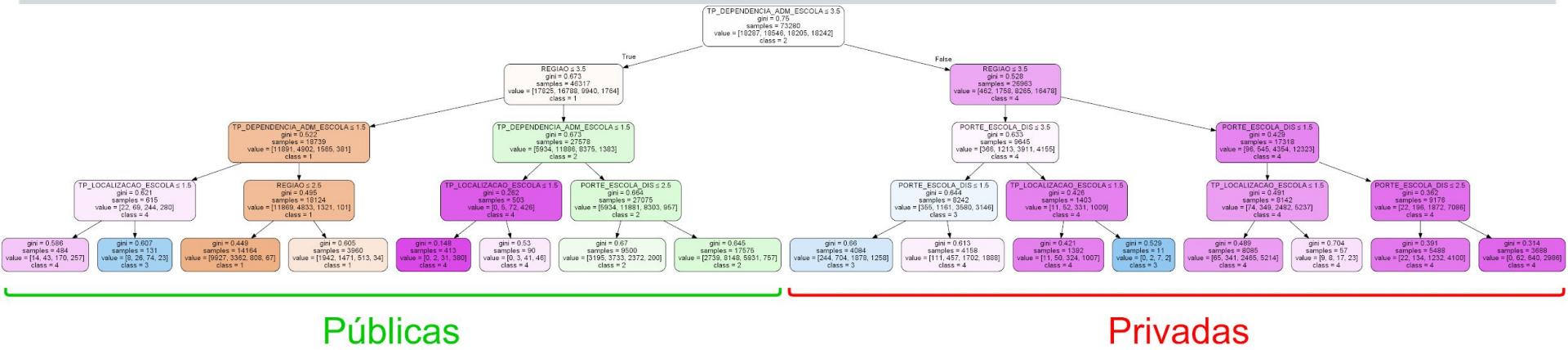
# Realiza a predição a partir do conjunto de teste
y_pred = clf.predict(X_test)

# Apresenta a acurácia do modelo
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

Accuracy: 0.5691724774731748

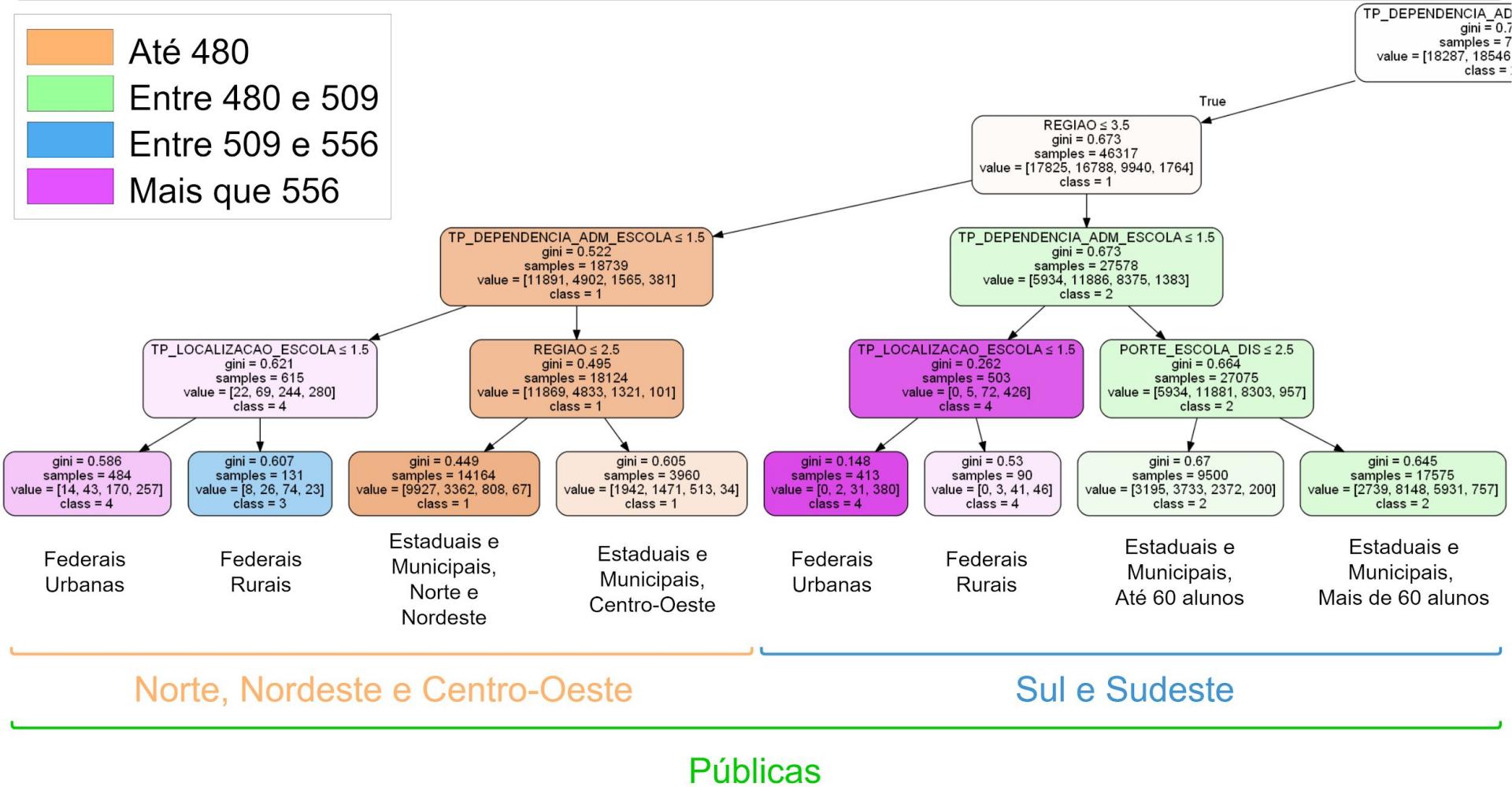
# Classificação

- Árvore de decisão:



# Classificação

- Até 480
- Entre 480 e 509
- Entre 509 e 556
- Mais que 556

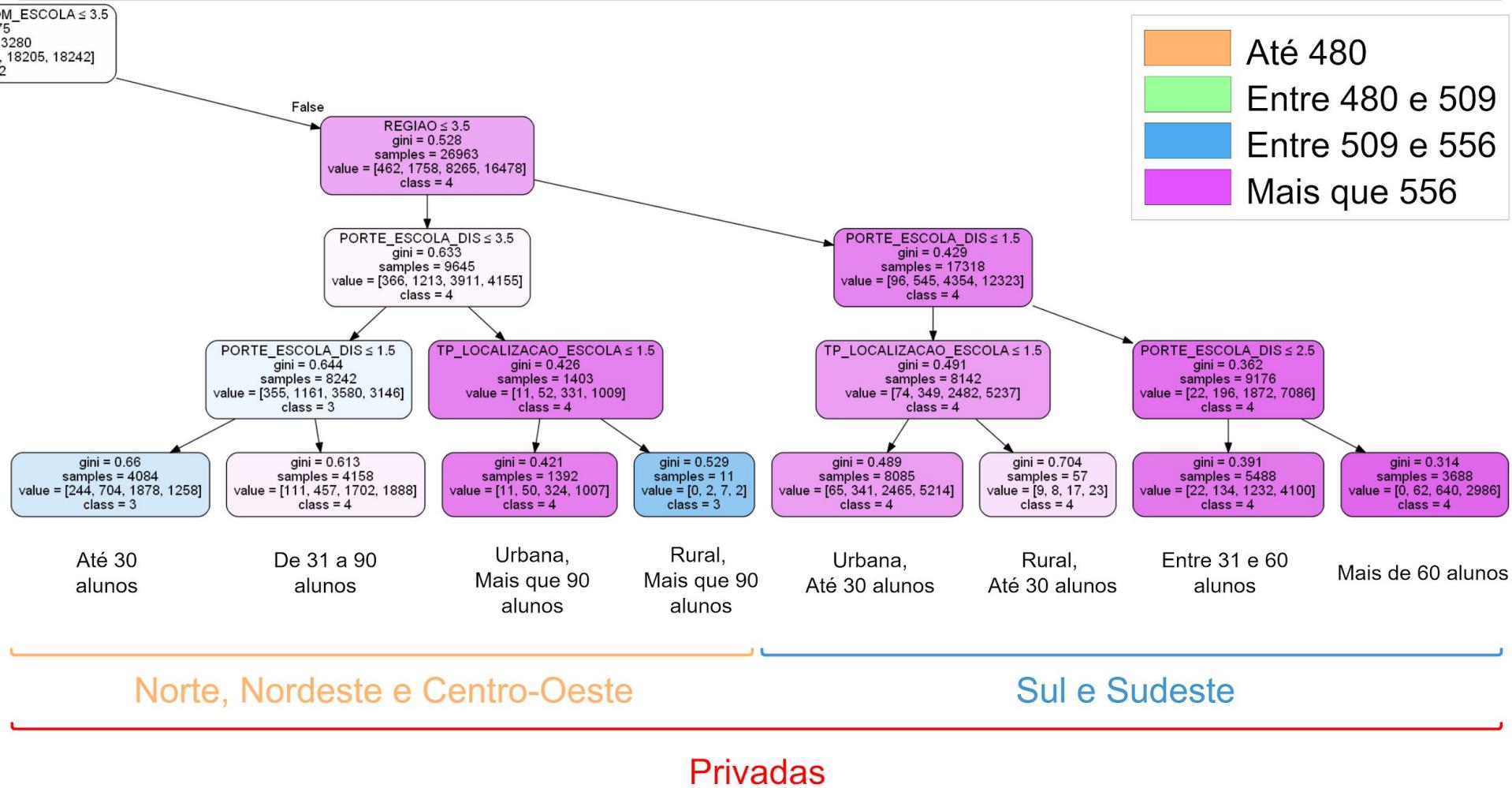


Norte, Nordeste e Centro-Oeste

Sul e Sudeste

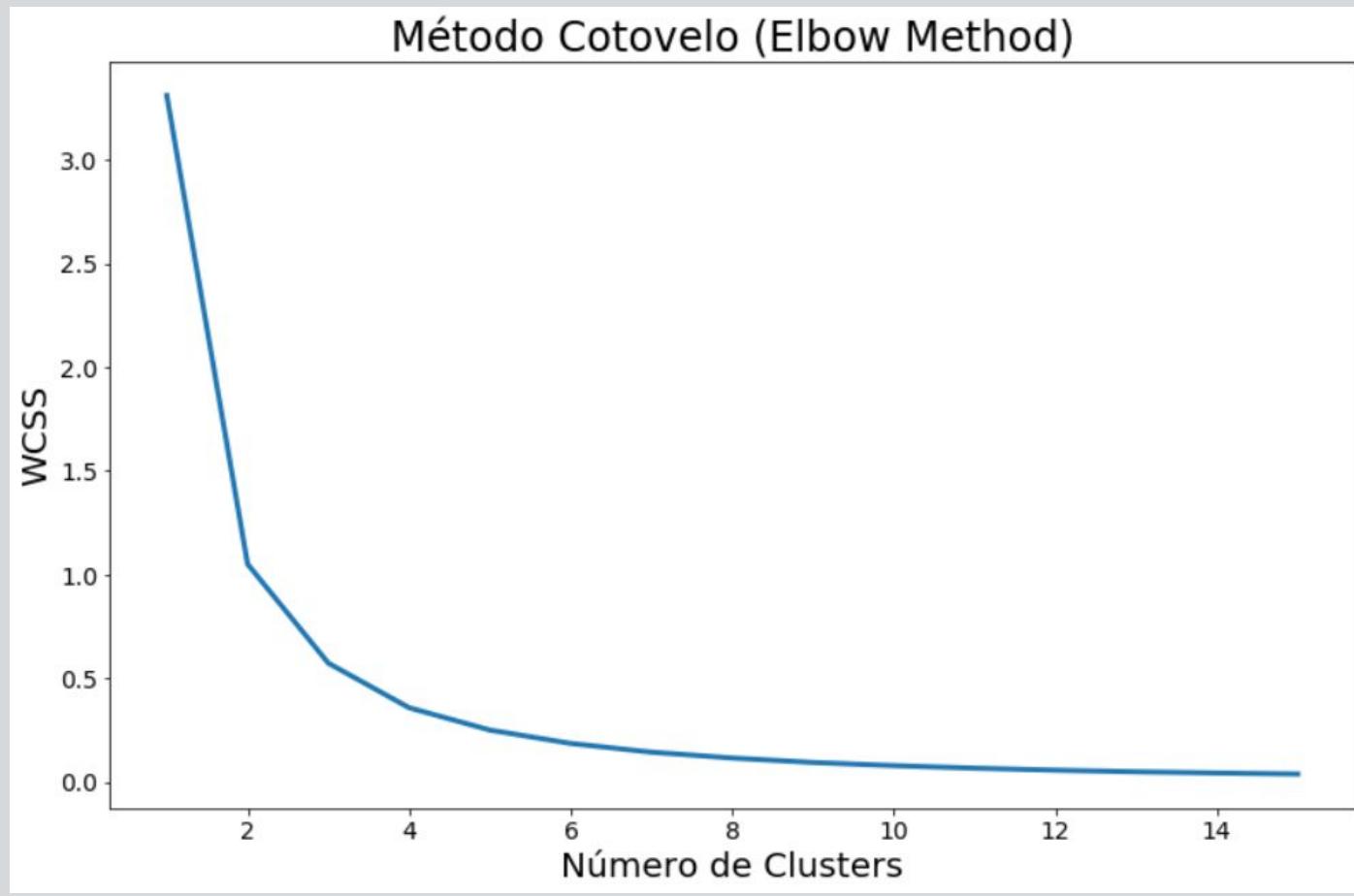
Públicas

# Classificação



# Clusterização K-mean

- Estimativa do número ótimo de clusters



# Clusterização K-mean

```
x = np.array(dataset).astype(float)

wcss = []
for i in range(1, 16):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, random_state=0)
    kmeans.fit(x)
    wcss.append(kmeans.inertia_)
```

```
def numero_otimo_clusters(wcss):
    x1, y1 = 1, wcss[0]
    x2, y2 = 15, wcss[len(wcss)-1]

    distancias = []
    for i in range(len(wcss)):
        x0 = i+2
        y0 = wcss[i]
        numerador = abs((y2-y1)*x0 - (x2-x1)*y0 + x2*y1 - y2*x1)
        denominador = ((y2 - y1)**2 + (x2 - x1)**2)**0.5
        distancias.append(numerador/denominador)

    return distancias.index(max(distancias)) + 1

print('Número ótimo de clusters:', numero_otimo_clusters(wcss))
```

Número ótimo de clusters: 3

# Clusterização K-mean

```
ds.head()
```

	TP_DEPENDENCIA_ADM_ESCOLA	TP_LOCALIZACAO_ESCOLA	MEDIA_TOT	REGIAO	PORTE_ESCOLA_DIS
67618		4	1	638.436	1
67619		4	1	641.634	1
67620		4	1	629.104	1
67621		4	1	625.290	1
67622		4	1	620.516	1

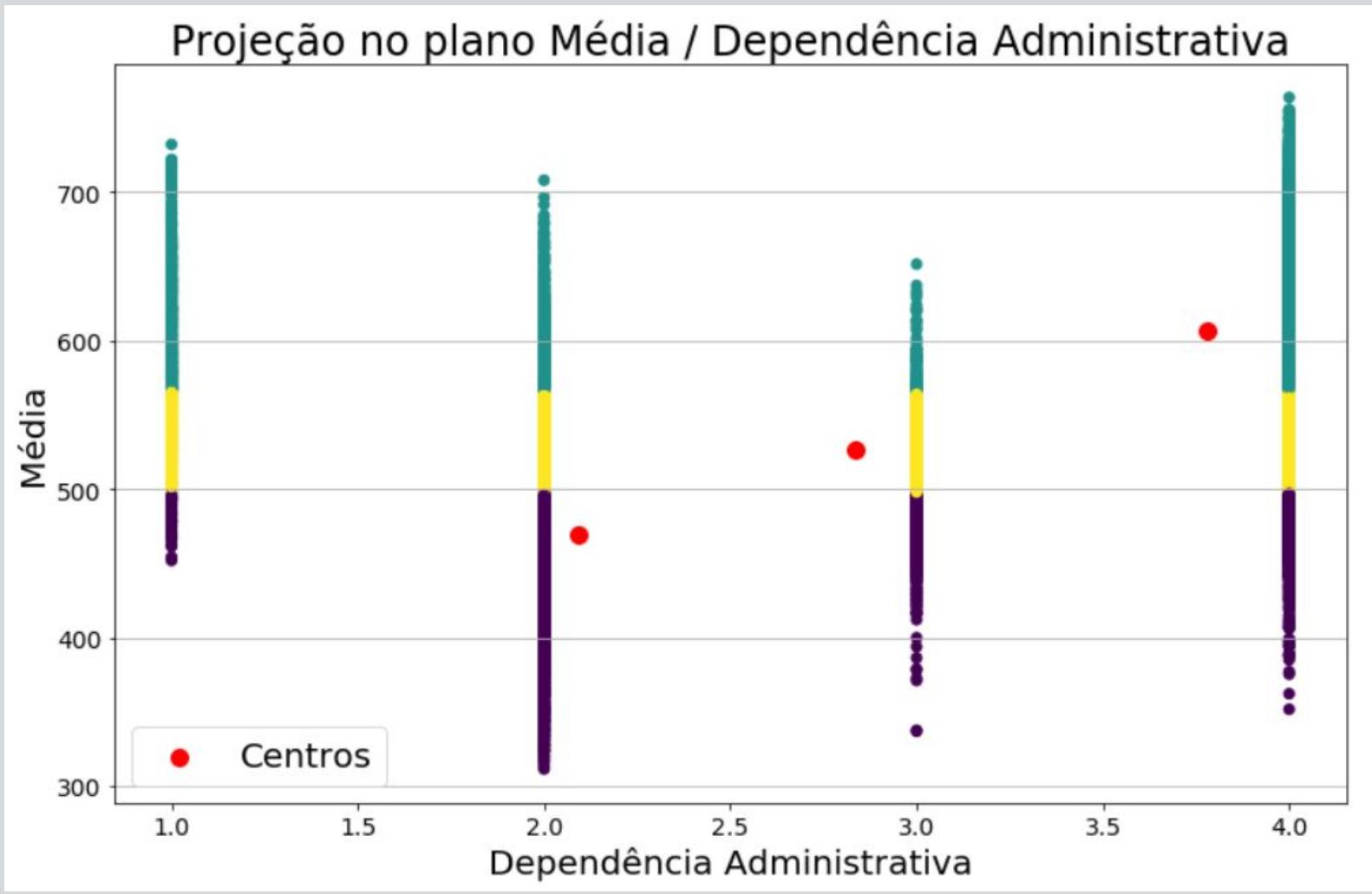
```
X = np.array(ds).astype(float)
```

```
# K-means
from sklearn.cluster import KMeans
model = KMeans(n_clusters=3, init='k-means++', n_init=10, random_state=0)
model.fit(X)
print(model.cluster_centers_)
```

```
[[ 2.09123781  1.06105072 469.88278402  3.0345262  2.96820324]
 [ 3.78224312  1.00669048 606.24077257  3.71136025  2.18864427]
 [ 2.83473552  1.02664987 526.39101567  3.66284635  2.58161209]]
```

- 1 → Federal
- 2 → Estadual
- 3 → Municipal
- 4 → Privada

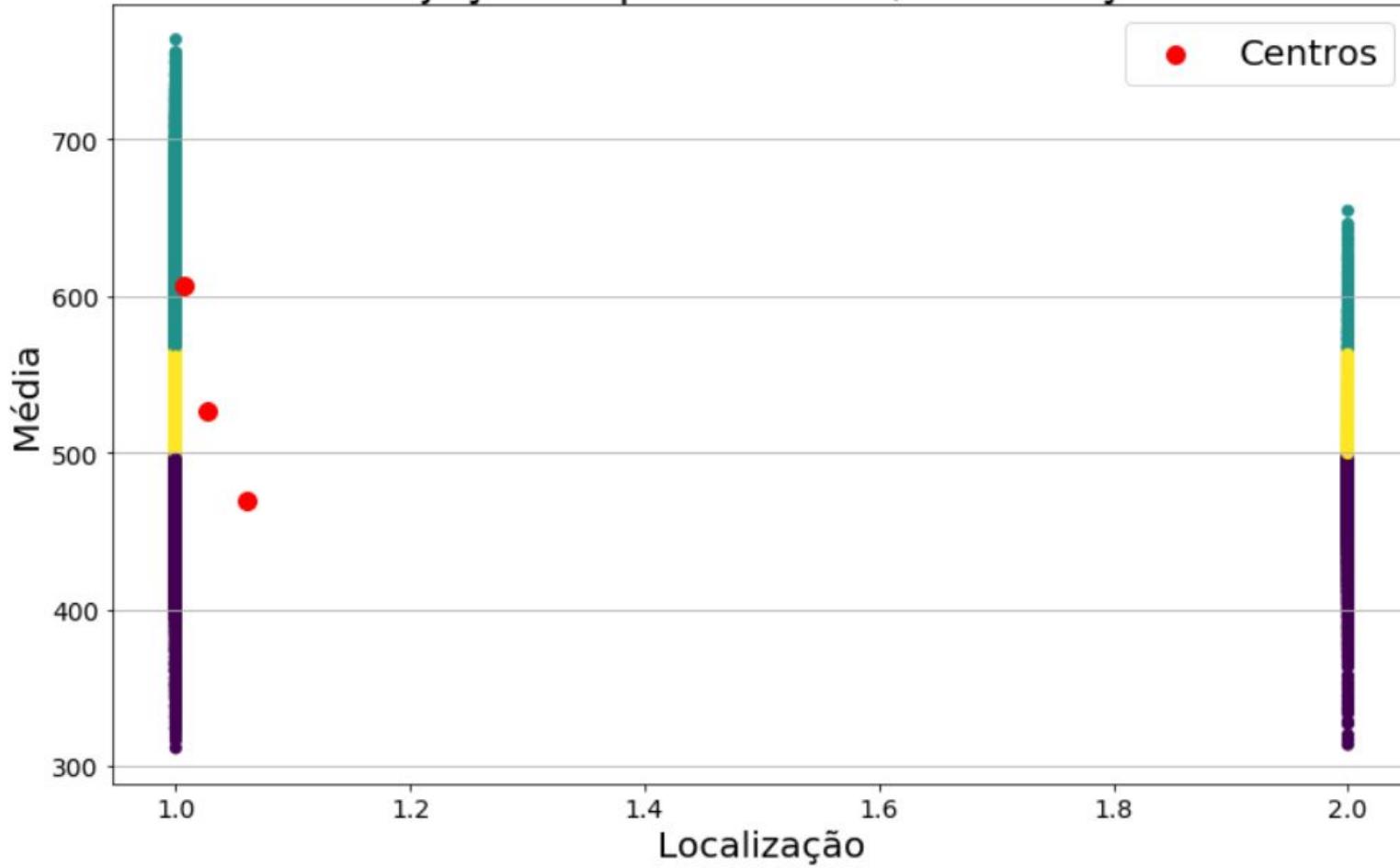
# Clusterização K-mean



# Clusterização K-mean

1 → Área Urbana  
2 → Área Rural

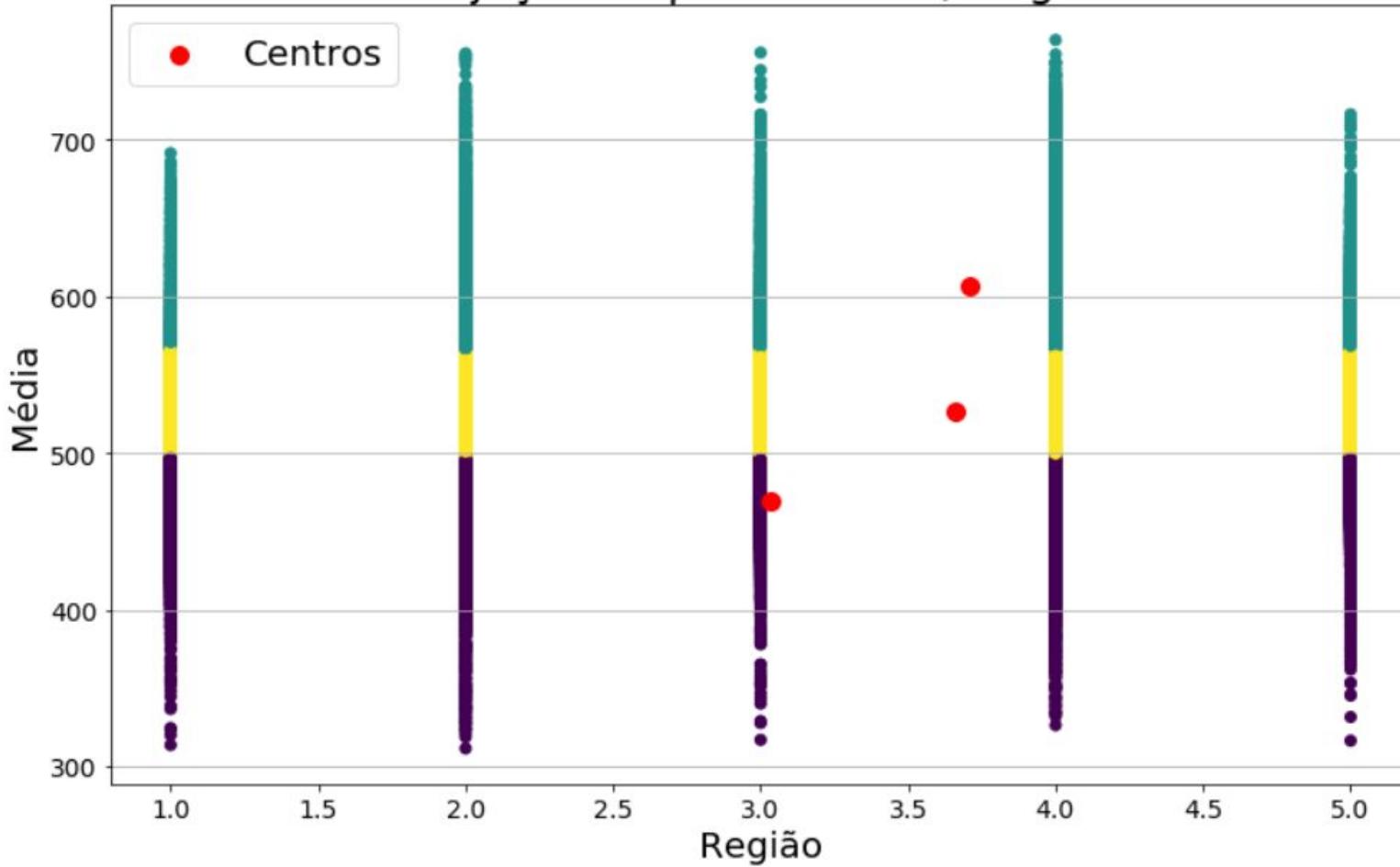
Projeção no plano Média / Localização



# Clusterização K-mean

- 1 → Norte
- 2 → Nordeste
- 3 → Centro-Oeste
- 4 → Sudeste
- 5 → Sul

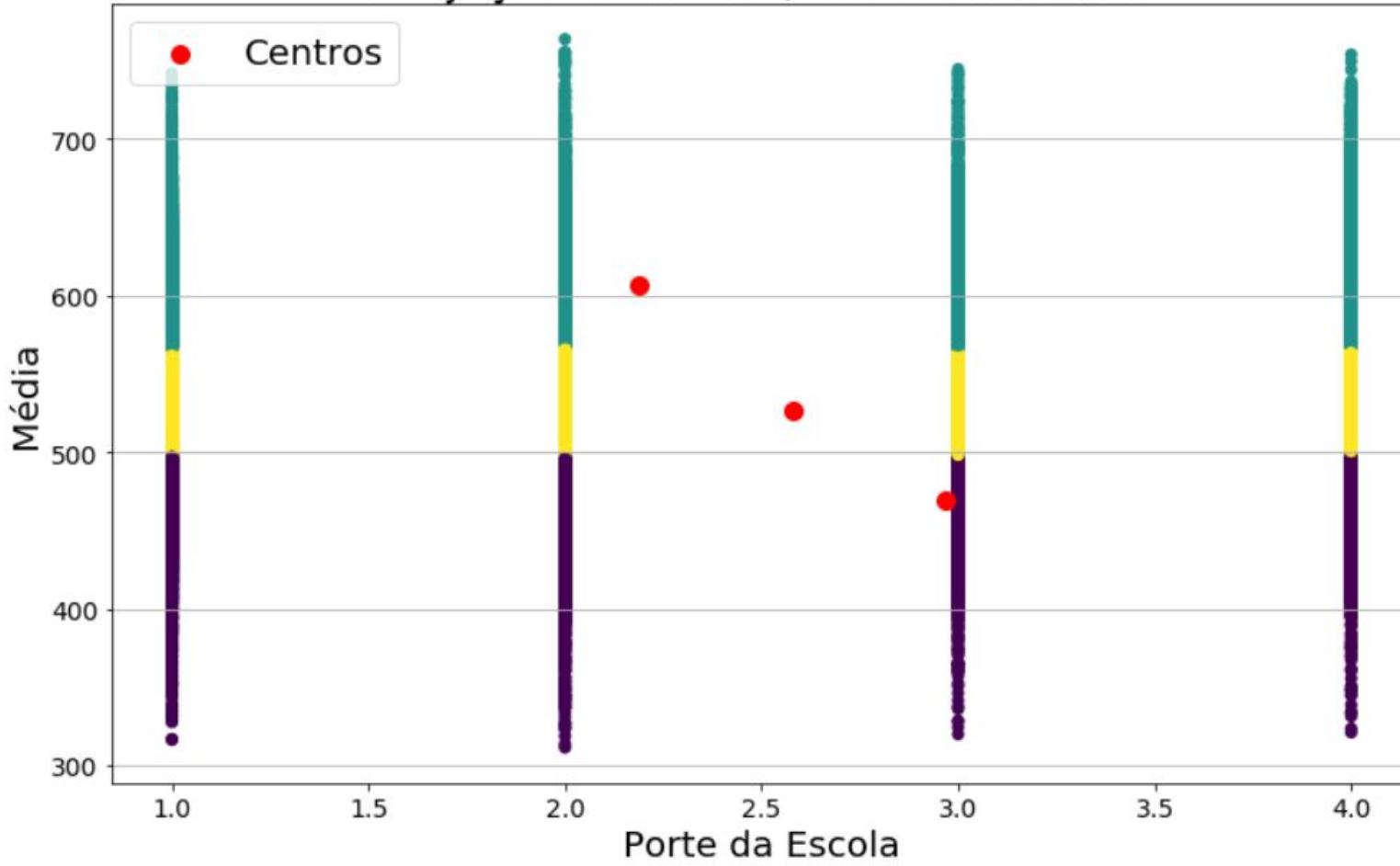
Projeção no plano Média / Região



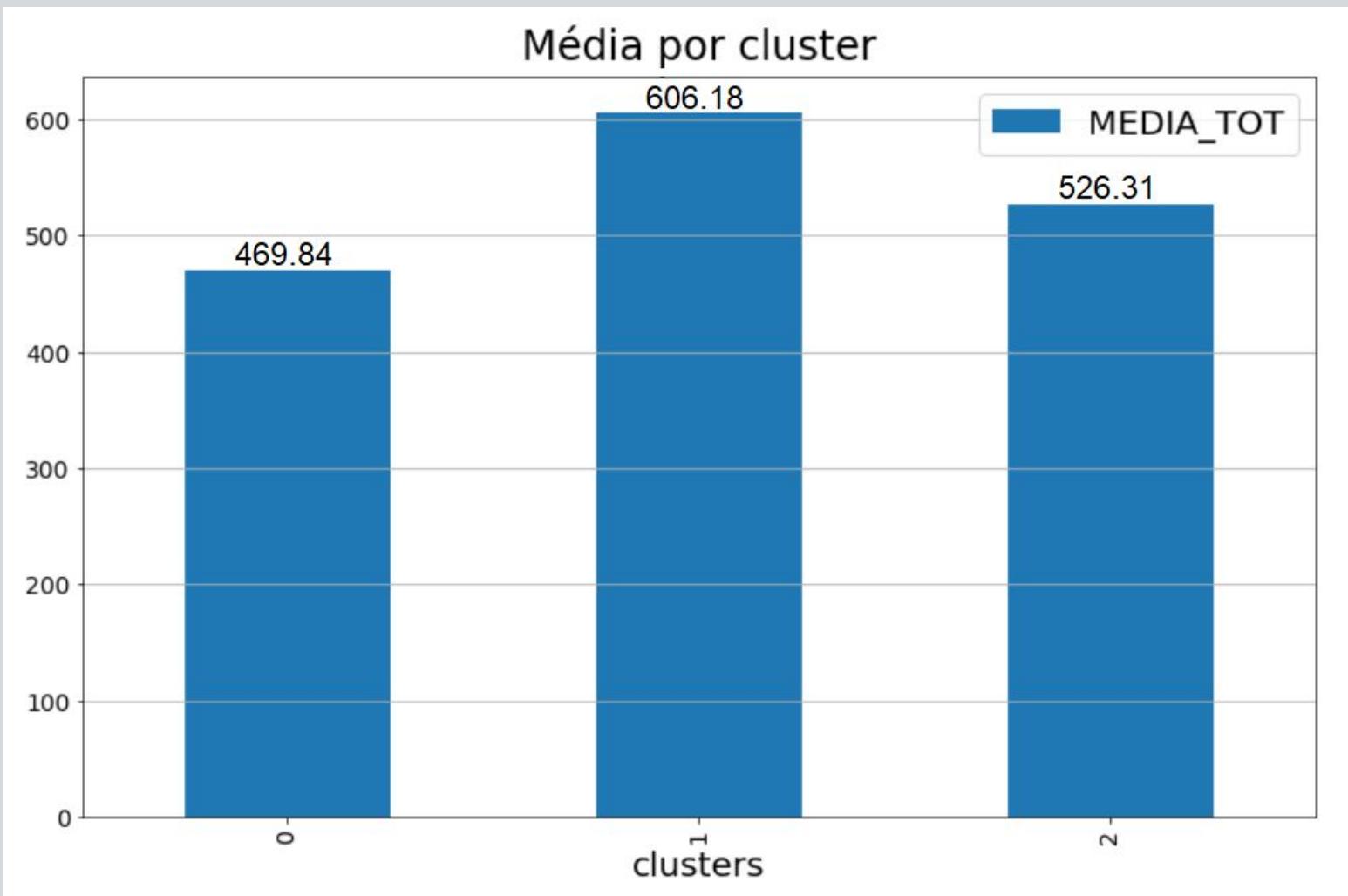
# Clusterização K-mean

- 1 → 01-30 alunos
- 2 → 31-60 alunos
- 3 → 31-90 alunos
- 4 → + 90 alunos

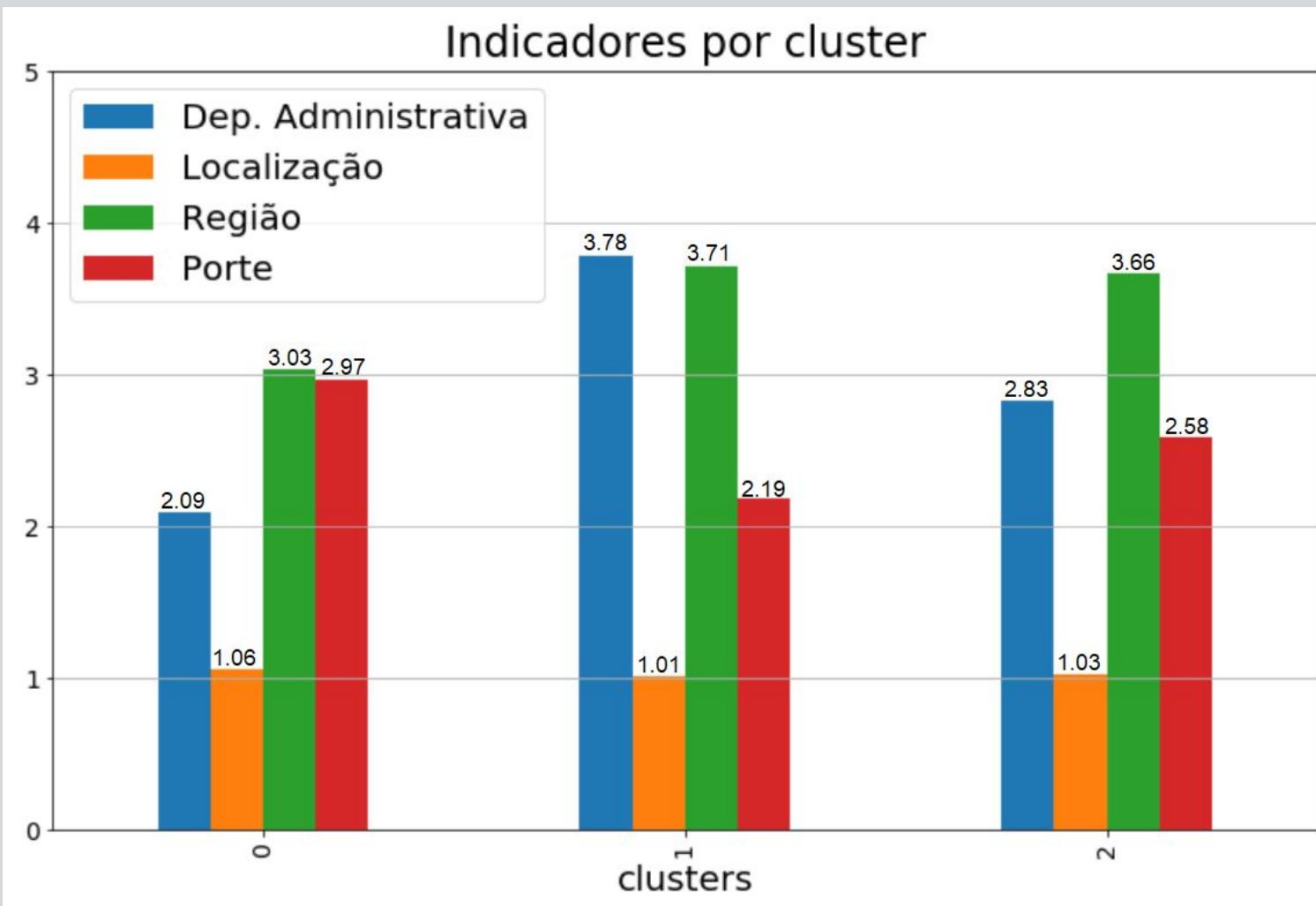
Projeção no Média / Porte da Escola



# Clusterização K-mean



# Clusterização K-mean



# Classificação

## - K vizinhos:

```
from sklearn.neighbors import KNeighborsClassifier  
  
model = KNeighborsClassifier(n_neighbors=4)  
  
# Treina o modelo utilizando o conjunto de treinamento  
model.fit(X_train,y_train)  
  
# Prediz a resposta para o conjunto de dados de teste  
y_pred = model.predict(X_test)  
  
# Apresenta a acurácia do modelo  
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

Accuracy: 0.5964593880345146

**Anterior: 0.505**

# Classificação

## - Naive Bayes:

```
# Importa o classificador
from sklearn.naive_bayes import GaussianNB

# Cria o classificador
gnb = GaussianNB()

# Treina o modelo usando o conjunto de treinamento
gnb.fit(X_train, y_train)

# Prediz a resposta para o conjunto de teste
y_pred = gnb.predict(X_test)

# Acurácia do Modelo; com que frequência o classificador está correto?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

Accuracy: 0.577864807208584

**Anterior: 0.547**

# Classificação

## - Random Forest:

```
from sklearn.ensemble import RandomForestClassifier

# Cria o classificador
clfRF=RandomForestClassifier(n_estimators=100)

# Treina o modelo usando os conjuntos de treinamento
clfRF.fit(X_train,y_train)

# Realiza a predição utilizando o conjunto de teste
y_pred=clfRF.predict(X_test)

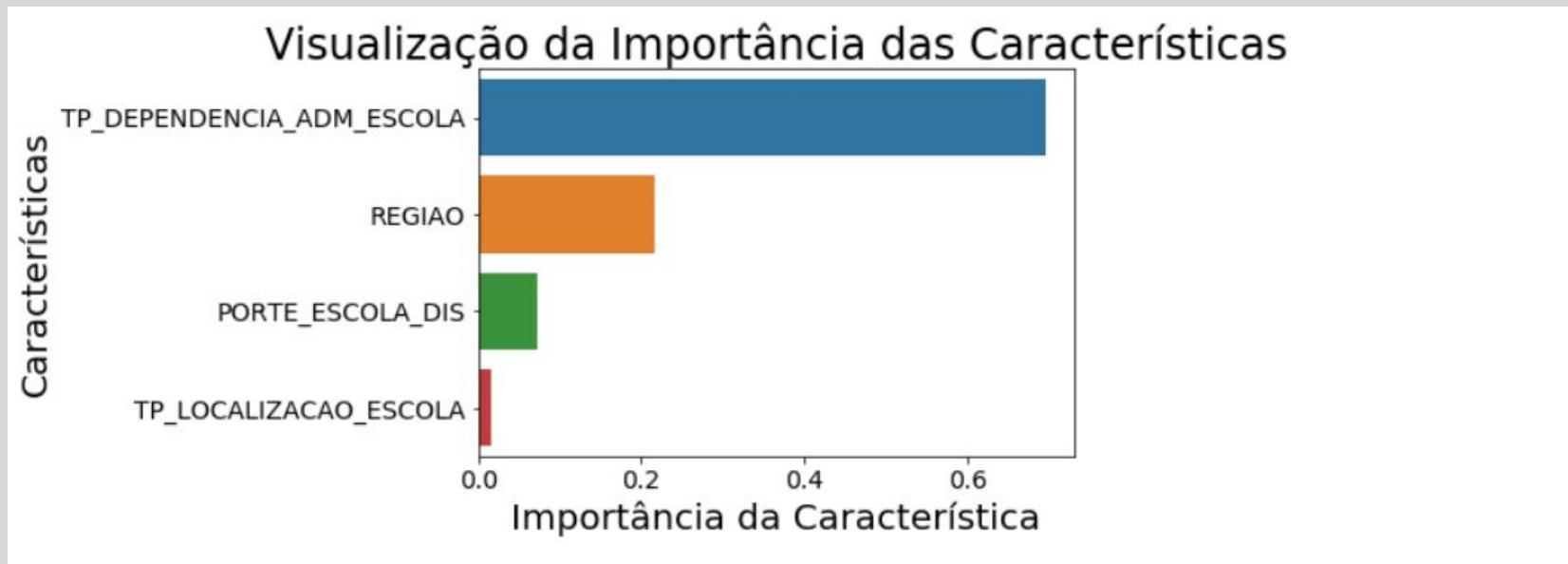
# Apresenta a acurácia do modelo
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

Accuracy: 0.6627503422803833

**Anterior: 0.578**

# Classificação

- Importância das Características:



# Classificação

- Árvore de decisão:

```
# Cria o objeto classificador para a árvore de decisão
clf = DecisionTreeClassifier(max_depth=4)

# Traina o objeto classificador
clf = clf.fit(X_train,y_train)

# Realiza a predição a partir do conjunto de teste
y_pred = clf.predict(X_test)

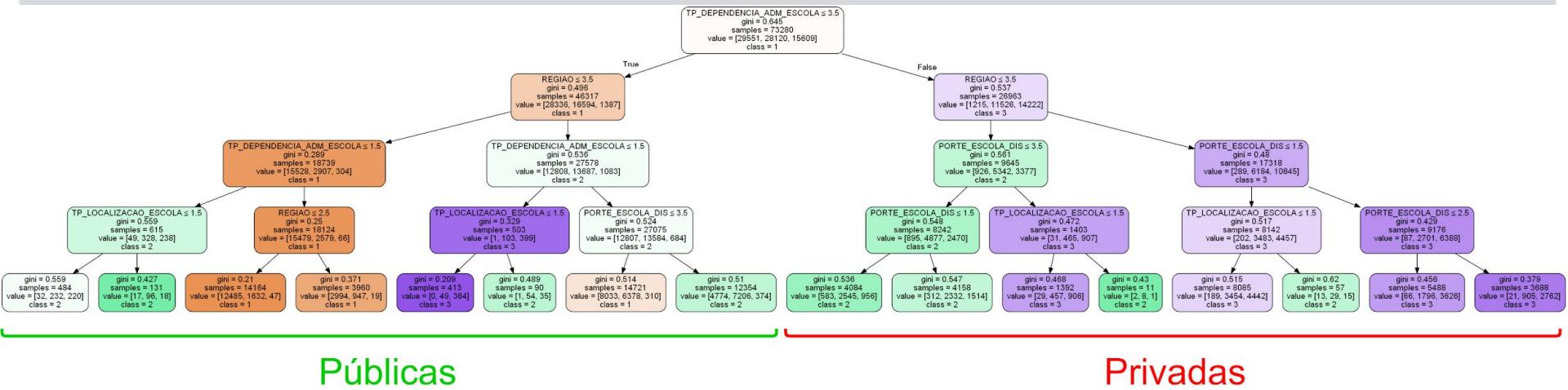
# Apresenta a acurácia do modelo
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

Accuracy: 0.6569554557901105

**Anterior: 0.569**

# Classificação

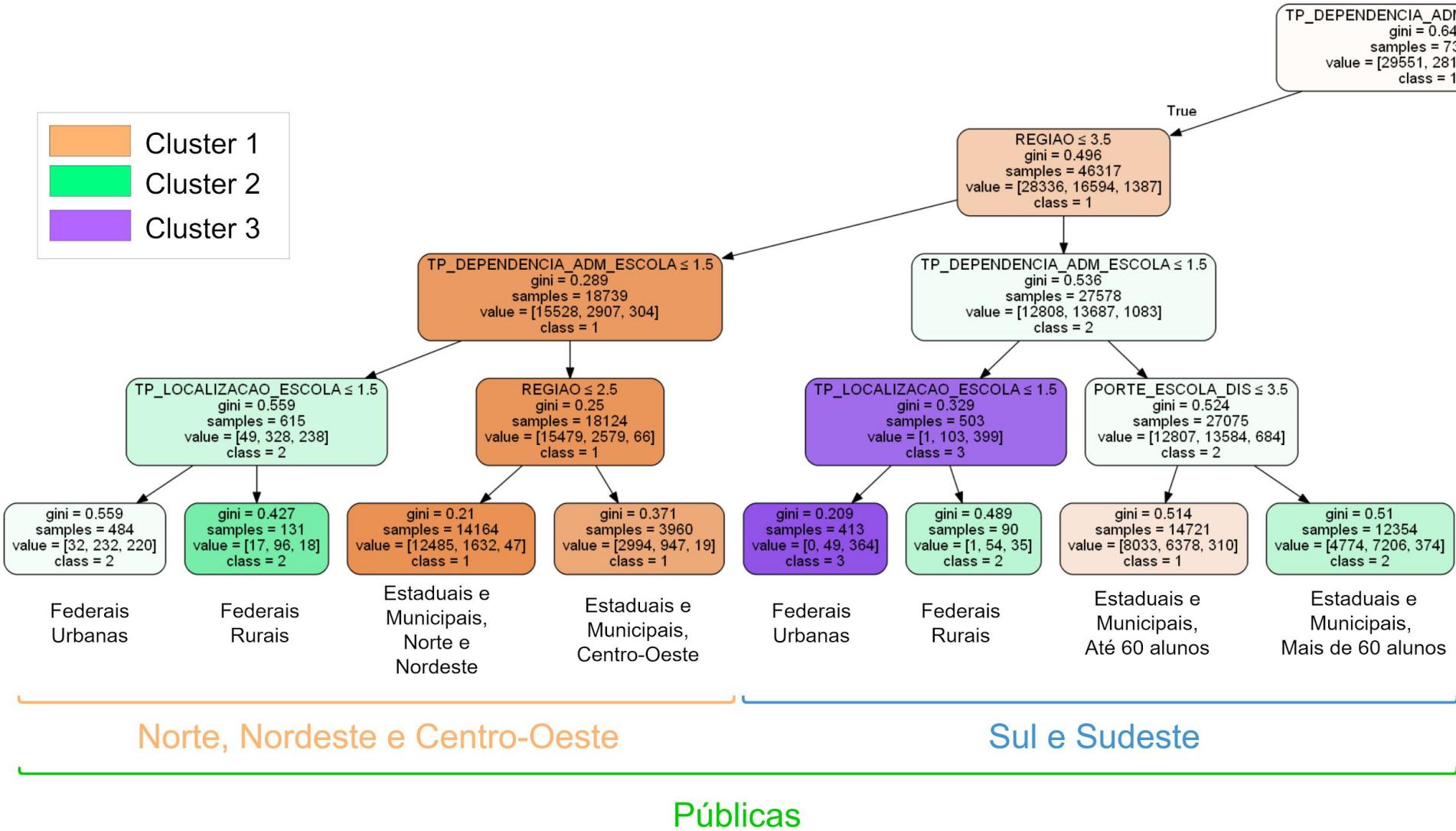
- Árvore de decisão:



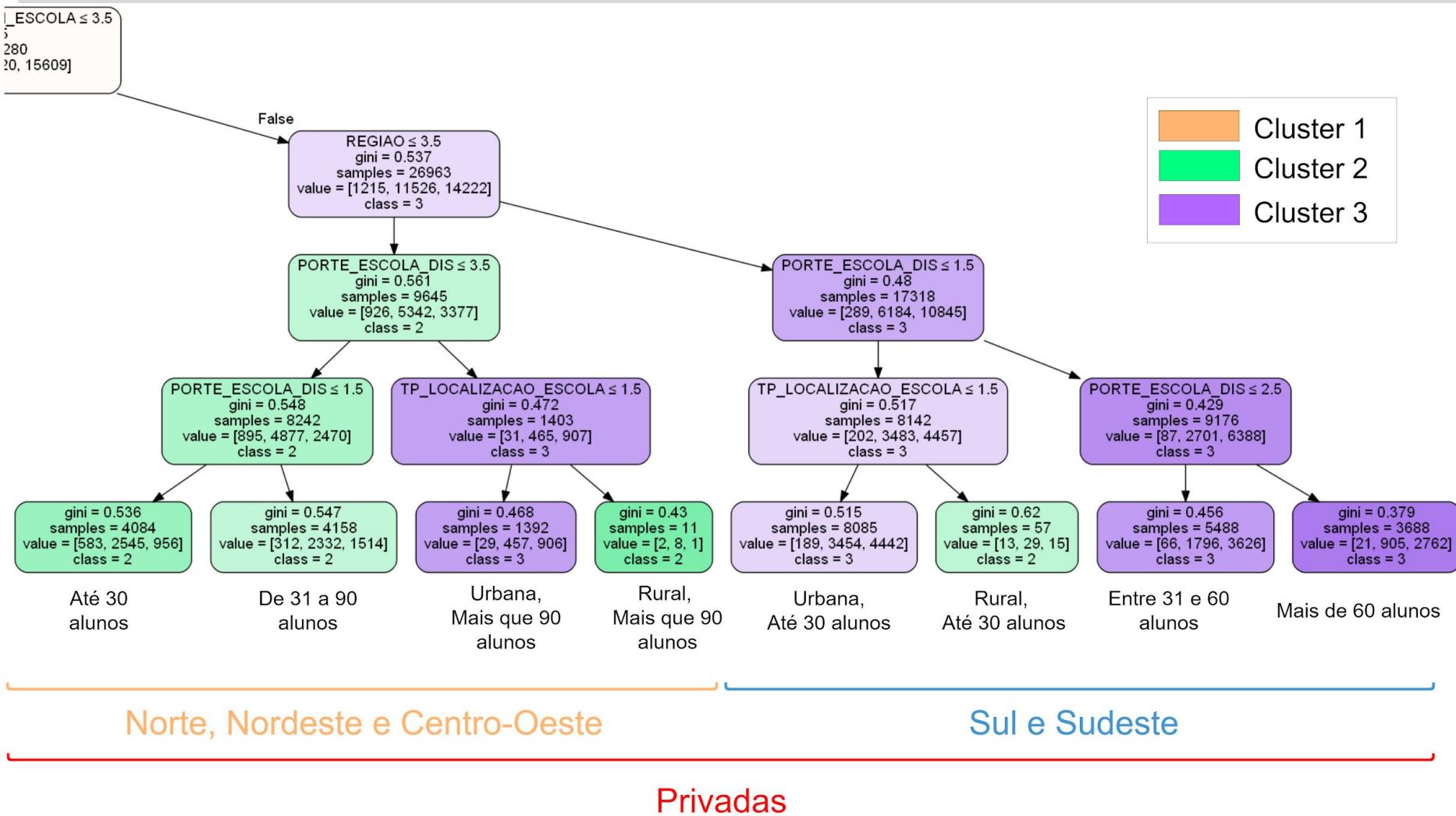
Públicas

Privadas

# Classificação



# Classificação



# Conclusões

- A análise permite apenas observar uma tendência entre os dados;
- A natureza dos dados exige cuidado na aferição de resultados;
- A análise a partir da nota média fornece um panorama razoável, mas abstrai diferenças relacionados a fatores internos;
- A proposta do ENEM não é ranquear os estados e as regiões pelo desempenho, mas essa análise permite avaliar características das políticas públicas em diferentes níveis.

# Referência

INSTITUTO NACIONAL DE ESTUDOS E PESQUISAS EDUCACIONAIS ANÍSIO TEIXEIRA. Microdados do Enem por Escola. Brasília: Inep, 2019. Disponível em: <<http://portal.inep.gov.br/web/guest/microdados>>. Acesso em: 01 nov. 2019.