Aula_regressao

June 11, 2025

1 Aula regressão

1.1 Regressão linear Simples

- Vamos modelar uma relação entre horas de estudo e nota na prova de alunos mas antes, vamos entender em dois modelos mais simples uma regressão linear.

[34]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
import pandas as pd
import seaborn as sns
```

[3]:
```
X = np.array([1,2,3,4,5,6,7,8,9,10]).reshape(-1, 1)
y = np.array([50,55,65,70,75,78,85,88,90,95])
```

[4]:
```
X
```

[4]:
```
array([[ 1],
       [ 2],
       [ 3],
       [ 4],
       [ 5],
       [ 6],
       [ 7],
       [ 8],
       [ 9],
       [10]])
```

[5]:
```
y
```

[5]:
```
array([50, 55, 65, 70, 75, 78, 85, 88, 90, 95])
```

[8]:
```
modelo = LinearRegression()
modelo.fit(X, y)
y_pred = modelo.predict([[1]])
print(y_pred)
```

```
[52.92727273]
```

[9]:
```
modelo = LinearRegression()
modelo.fit(X, y)
y_pred = modelo.predict(X)
print(y_pred)
```

```
[52.92727273 57.85454545 62.78181818 67.70909091 72.63636364 77.56363636
 82.49090909 87.41818182 92.34545455 97.27272727]
```

[10]:
```
print(f"Equação: y = {modelo.intercept_:.2f} + {modelo.coef_[0]:.2f}x")
```

```
Equação: y = 48.00 + 4.93x
```

[15]:
```
y_mostra = modelo.intercept_ + modelo.coef_ * 20
y_mostra
```

[15]:
```
array([146.54545455])
```

[16]:
```
plt.scatter(X, y, color='blue', label='Dados Reais')
plt.plot(X, y_pred, color='red', label='Reta de Regressão')
plt.xlabel("Horas de Estudo")
plt.ylabel("Nota na Prova")
plt.title("Regressão Linear Simples")
plt.legend()
plt.grid(True)
plt.show()
```

[17]:
```
print(f"Equação: y = {modelo.intercept_:.2f} + {modelo.coef_[0]:.2f}x")
```

```
Equação: y = 48.00 + 4.93x
```

[18]:
```
from sklearn.linear_model import LogisticRegression
```

[20]:
```
# Criando dados sintéticos
df2 = pd.read_csv('simulacao_aprovacao.csv')
df2
```

[20]:

	horas_estudo	faltas	participacao	aprovado
0	9	0	baixa	1
1	4	4	baixa	0
2	0	4	alta	0
3	1	5	alta	0
4	9	5	baixa	0
..	…	…	…	…
195	7	4	baixa	0
196	7	5	alta	1
197	6	3	alta	1

```
198             9       5         alta         1
199             2       1         alta         1

[200 rows x 4 columns]
```

[22]:
```
# Codificando variável categórica
from sklearn.preprocessing import LabelEncoder
le_part = LabelEncoder()
df2['participacao_cod'] = le_part.fit_transform(df2['participacao'])
df2
```

[22]:
```
     horas_estudo  faltas participacao  aprovado  participacao_cod
0               9       0        baixa         1                 1
1               4       4        baixa         0                 1
2               0       4         alta         0                 0
3               1       5         alta         0                 0
4               9       5        baixa         0                 1
..            ...     ...          ...       ...               ...
195             7       4        baixa         0                 1
196             7       5         alta         1                 0
197             6       3         alta         1                 0
198             9       5         alta         1                 0
199             2       1         alta         1                 0

[200 rows x 5 columns]
```

[23]:
```
# Treinamento

X = df2[['horas_estudo', 'faltas', 'participacao_cod']]
y = df2['aprovado']
```

[24]:
```
X
```

[24]:
```
     horas_estudo  faltas  participacao_cod
0               9       0                 1
1               4       4                 1
2               0       4                 0
3               1       5                 0
4               9       5                 1
..            ...     ...               ...
195             7       4                 1
196             7       5                 0
197             6       3                 0
198             9       5                 0
199             2       1                 0

[200 rows x 3 columns]
```

[25]:
```
y
```

[25]:
```
0      1
1      0
2      0
3      0
4      0
      ..
195    0
196    1
197    1
198    1
199    1
Name: aprovado, Length: 200, dtype: int64
```

[28]:
```
# Criando o modelo

modelo_log = LogisticRegression()
modelo_log.fit(X, y)

df2['prob_aprovado'] = modelo_log.predict_proba(X)[:,1]
df2['classe_prevista'] = modelo_log.predict(X)
```

[29]:
```
df2['prob_aprovado']
```

[29]:
```
0      0.996522
1      0.054662
2      0.172544
3      0.103716
4      0.330762
         …
195    0.310138
196    0.874923
197    0.978260
198    0.964854
199    0.973844
Name: prob_aprovado, Length: 200, dtype: float64
```

[30]:
```
df2['classe_prevista']
```

[30]:
```
0      1
1      0
2      0
3      0
4      0
      ..
195    0
```

```
196    1
197    1
198    1
199    1
Name: classe_prevista, Length: 200, dtype: int64
```

[31]:
```
df2
```

[31]:

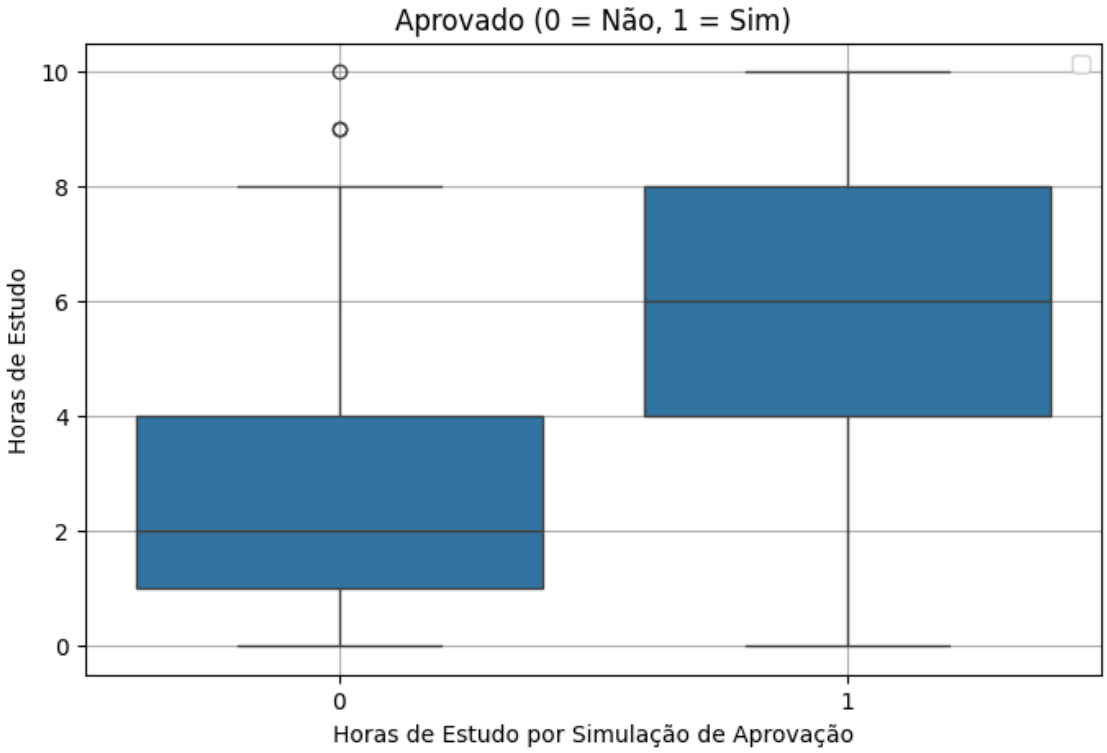
	horas_estudo	faltas	…	prob_aprovado	classe_prevista
0	9	0	…	0.996522	1
1	4	4	…	0.054662	0
2	0	4	…	0.172544	0
3	1	5	…	0.103716	0
4	9	5	…	0.330762	0
..	…	…	…	…	…
195	7	4	…	0.310138	0
196	7	5	…	0.874923	1
197	6	3	…	0.978260	1
198	9	5	…	0.964854	1
199	2	1	…	0.973844	1

[200 rows x 7 columns]

[35]:
```
# simulaçao das horas

plt.figure(figsize=(8,5))
sns.boxplot(x='aprovado', y='horas_estudo', data=df2)
plt.xlabel("Horas de Estudo por Simulação de Aprovação")
plt.ylabel("Horas de Estudo")
plt.title("Aprovado (0 = Não, 1 = Sim)")
plt.legend()
plt.grid(True)
plt.show()
```

```
/var/folders/yq/pq2twqh913s0mr445r3_jz7h0000gn/T/ipykernel_59833/3208887324.py:8
: UserWarning: No artists with labels found to put in legend.  Note that artists
whose label start with an underscore are ignored when legend() is called with no
argument.
  plt.legend()
```



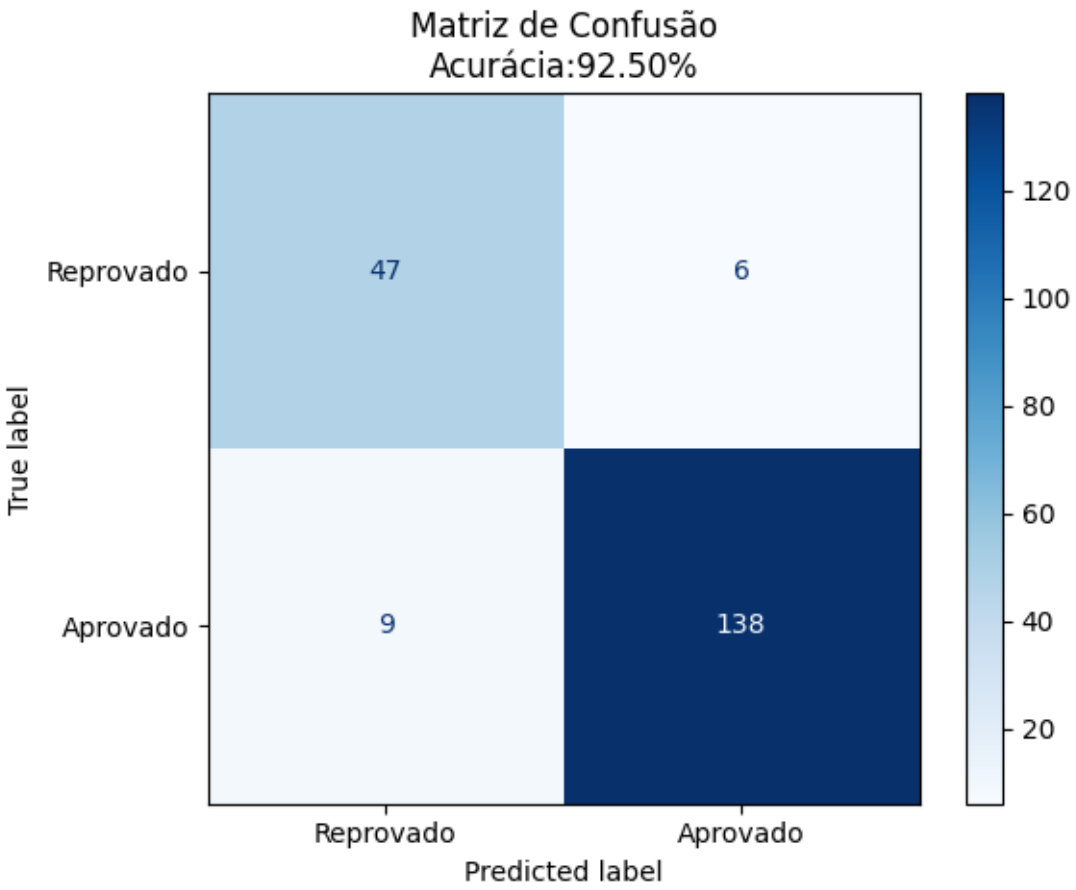
[36]:
```
# Criando matriz de confusão

from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay,␣
 ↪accuracy_score

cm = confusion_matrix(df2['aprovado'], df2['classe_prevista'])
acc = accuracy_score(df2['aprovado'], df2['classe_prevista'])

disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['Reprovado',␣
 ↪'Aprovado'])
plt.figure(figsize=(6,6))
disp.plot(cmap='Blues', values_format='d')
plt.title(f'Matriz de Confusão\nAcurácia:{acc:.2%}')
plt.grid(False)
plt.show()
```

```
<Figure size 600x600 with 0 Axes>
```



[38]:
```
# Relatório completo de precisão

from sklearn.metrics import classification_report
relatorio = classification_report(df2['aprovado'], df2['classe_prevista'],␣
 ↪target_names=['Reprovado', 'Aprovado'], output_dict=True)
relatorio_df = pd.DataFrame(relatorio).transpose()
relatorio_df
```

[38]:

	precision	recall	f1-score	support
Reprovado	0.839286	0.886792	0.862385	53.000
Aprovado	0.958333	0.938776	0.948454	147.000
accuracy	0.925000	0.925000	0.925000	0.925
macro avg	0.898810	0.912784	0.905419	200.000
weighted avg	0.926786	0.925000	0.925646	200.000

[39]:
```
2*((0.839286*0.886792)/(0.839286+0.886792))
```

[39]: 0.8623852578064259

1.2 Exemplo 1: Você é cientista de dados em uma empresa de serviços por assinatura.

Desafio: Com base nos dados apresentados no dataset, responda:
Qual cliente tem maior risco de cancelamento?
Quais variáveis mais influenciam esse risco?
O que você recomendaria para reter esse cliente?

Conjunto de Dados Simulado:
Cliente A: 24 meses, sem atraso, plano premium, usa frequentemente
Cliente B: 5 meses, 2 atrasos, plano básico, pouco engajado
Cliente C: 10 meses, 1 reclamação, plano intermediário, uso médio

Tarefa: Defina qual cliente você abordaria e por quê.

[ ]: