ml-energia-solar

June 28, 2025

MACHINE LEARNING PARA REGRESSÃO

Este notebook apresenta um exercício prático de **Machine Learning para Regressão de Dados** utilizando as bibliotecas: **Pandas**, **Scikit-Learn** e **Matplotlib** do Python.

Será utilizado um conjunto de dados formatado como texto delimitado (CSV) contendo dados ambientais. O objetivo é realizar a previsão da energia solar disponível na superfície terrestre usando dados climáticos como variáveis preditoras. Serão consideradas as seguintes técnicas de treinamento de modelos de Machine Learning:

- Ingestão de dados.
- Separação dos dados.
- Treinamento do modelo de Machine Learning.
- Avaliação estatística dos resultado.
- Avaliação gráfica dos resultados.

1 - Instalação das Bibliotecas

[ ]:
```
# Instalação do Pandas para manipulação de dados.
!pip install pandas

# Instalação do Scikit-Learn para modelagem de machine learning.
!pip install scikit-learn

# Instalação do Matplotlib para criação de gráficos.
!pip install matplotlib
```

2 - Importação das Bibliotecas

[ ]:
```
# Usada para manipulação de dados.
import pandas

# Usadas para construir os modelos de machine learning.
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.linear_model import LinearRegression

# Usada para separação dos dados.
from sklearn.model_selection import train_test_split

# Usada para avaliar o desempenho do machine learning.
```

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# Usadas para construir gráficos.
import matplotlib.pyplot as pyplot
from matplotlib.ticker import MultipleLocator
```

3 - Importação dos Dados

O conjunto de dados utilizado contém informações meteorológicas medidas na superfície terrestre de uma cidade do estado de São Paulo (Botucatu). Ele contém as seguintes colunas:

- **ID**: Identificação única do registro de dados.
- **Ano**: Ano de coleta dos dados.
- **Mês**: Mês de coleta dos dados.
- **Dia**: Dia de coleta dos dados.
- **DJ**: O dia juliano, que representa a contagem de dias desde o início do ano.
- **Fotoperiodo**: O número de horas de luz solar nesse dia específico.
- **TAR**: Temperatura do ar em graus Celsius.
- **UR**: Umidade relativa do ar, expressa em porcentagem.
- **Vento-2M**: Velocidade do vento a 2 metros do solo, em metros por segundo.
- **Extraterrestre**: Radiação solar extraterrestre em MJ por metro quadrado.
- **Energia Solar**: Energia solar medida na superfície terrestre em MJ por metro quadrado.

[ ]:
```
# Define o caminho do arquivo de texto delimitado.
arquivo = "Energia Solar.csv"

# Carrega o arquivo de texto delimitado.
df = pandas.read_csv(arquivo, sep=";", decimal=".", encoding="utf-8")

# Exibe os dados carregados.
display(df.head(15))
```

4 - Separação dos Dados

[ ]:
```
# Divide os dados para treinamento e validação.
df_treino, df_teste = train_test_split(df, test_size=0.2, random_state=42)

# Separa as variáveis independentes (X) e dependente (y) de treinamento.
X_treino = df_treino[['Fotoperiodo', 'TAR', 'UR', 'Extraterrestre']]
Y_treino = df_treino['Energia Solar']

# Separa as variáveis independentes (X) e dependente (y) de validação.
X_teste = df_teste[['Fotoperiodo', 'TAR', 'UR', 'Extraterrestre']]
Y_teste = df_teste['Energia Solar']
```

5 - Treinamento do Modelo de Machine Learning

[ ]:
```
# Cria o modelo de Regressão Linear.
modelo_RL = LinearRegression()
```

```
# Ajusta o modelo de Regressão Linear aos dados.
modelo_RL.fit(X_treino, Y_treino)

# Cria o modelo de Gradient Boosting.
modelo_GB = GradientBoostingRegressor(random_state=42)

# Ajusta o modelo de Gradient Boosting aos dados.
modelo_GB.fit(X_treino, Y_treino)
```

6 - Avaliação Estatística dos Resultados

[ ]:
```
# Calcula as métricas estatísticas para avaliar o modelo de Regressão Linear.
RL_mbe = mean_absolute_error(Y_teste, modelo_RL.predict(X_teste))
RL_rmse = mean_squared_error(Y_teste, modelo_RL.predict(X_teste))
RL_r2 = r2_score(Y_teste, modelo_RL.predict(X_teste))

# Exibe os resultados para o modelo Regressão Linear (RL).
print(f"Resultados do Modelo de Regressão Linear:")
print(f"MBE: {RL_mbe:.2f}")
print(f"RMSE: {RL_rmse:.2f}")
print(f"R²: {RL_r2:.2f}")

# Calcula as métricas estatísticas para avaliar o modelo de Gradient Boosting.
GB_mbe = mean_absolute_error(Y_teste, modelo_GB.predict(X_teste))
GB_rmse = mean_squared_error(Y_teste, modelo_GB.predict(X_teste))
GB_r2 = r2_score(Y_teste, modelo_GB.predict(X_teste))

# Exibe os resultados para o modelo de Gradient Boosting (GB).
print(f"Resultados do Modelo de Gradient Boosting:")
print(f"MBE: {GB_mbe:.2f}")
print(f"RMSE: {GB_rmse:.2f}")
print(f"R²: {GB_r2:.2f}")
```

7 - Avaliação Gráfica dos Resultados

[ ]:
```
# Gráfico de dispersão para o modelo de Regressão Linear.
pyplot.figure(figsize=(6, 6))
pyplot.scatter(Y_teste, modelo_RL.predict(X_teste), alpha=0.5, color='green')
pyplot.plot([Y_teste.min(), Y_teste.max()], [Y_teste.min(), Y_teste.max()],␣
↪'--k', linewidth=2)
pyplot.xlabel('Energia Solar - Estimada (MJ/m²)')
pyplot.ylabel('Energia Solar - Medida (MJ/m²)')
pyplot.title('Gráfico de Dispersão - Regressão Linear', weight='bold')

# Define os limites do eixo X e Y.
pyplot.xlim(0, 30)
```

```
pyplot.ylim(0, 30)

# Configura os intervalos nos eixos X e Y.
pyplot.gca().xaxis.set_major_locator(MultipleLocator(2))
pyplot.gca().yaxis.set_major_locator(MultipleLocator(2))

# Exibe o gráfico para o modelo de Regressão Linear.
pyplot.show()

# Gráfico de dispersão para o Gradient Boosting.
pyplot.figure(figsize=(6, 6))
pyplot.scatter(Y_teste, modelo_GB.predict(X_teste), alpha=0.5, color='blue')
pyplot.plot([Y_teste.min(), Y_teste.max()], [Y_teste.min(), Y_teste.max()],␣
 ↪'--k', linewidth=2)
pyplot.xlabel('Energia Solar - Estimada (MJ/m²)')
pyplot.ylabel('Energia Solar - Medida (MJ/m²)')
pyplot.title('Gráfico de Dispersão - Gradient Boosting', weight='bold')

# Define os limites do eixo X e Y.
pyplot.xlim(0, 30)
pyplot.ylim(0, 30)

# Configura os intervalos nos eixos X e Y.
pyplot.gca().xaxis.set_major_locator(MultipleLocator(2))
pyplot.gca().yaxis.set_major_locator(MultipleLocator(2))

# Exibe o gráfico para o Gradient Boosting.
pyplot.show()
```

8 - Consumo dos Modelos de Machine Learning

[ ]:
```
# Inserção manual dos valores para a previsão.
entradas = {
    'Fotoperiodo': 13.28607,
    'TAR': 27.37,
    'UR': 64.45,
    'Extraterrestre': 42.58281
}

# Cria um data frame com os valores inseridos manualmente.
df_entradas = pandas.DataFrame([entradas])

# Escolhe do modelo para previsão (RL ou GB).
escolha = 'GB'

# Faz as previsões usando o modelo escolhido.
if escolha == 'GB':
```

```
    estimativa = modelo_GB.predict(df_entradas)
    print(f"Energia Solar Estimada com Gradient Boosting: {estimativa[0]:.2f}␣
 ↪MJ/m²")

elif escolha == 'RL':
    estimativa = modelo_RL.predict(df_entradas)
    print(f"Energia Solar Estimada com Regressão Linear: {estimativa[0]:.2f} MJ/
 ↪m²")

else:
    print("Modelo escolhido inválido. Por favor, escolha: 'GB' ou 'RL'.")
```