Atividade Experimental 29/04/2025

*** Você deve fazer um relatório de análise dos resultados experimentais e entregar nesta tarefa.

Regressão linear
1. Utilizando todo o dataset Advertising.csv (em anexo):
a- Utilizando as bibliotecas pandas e statmodels o código a seguir faz a regressão linear (simples usando cada atributo, do dataset).
Execute o código e faça uma análise dos resultados de acordo com o que foi discutido nas últimas aulas.

```
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
import seaborn as sns

# Carrega o dataset
url                                                                    =
'https://raw.githubusercontent.com/selva86/datasets/master/Advertising.
csv'
df = pd.read_csv(url, index_col=0)

# Lista de atributos para analisar
features = ['TV', 'radio', 'newspaper']

# Estilo dos gráficos
sns.set(style="whitegrid")

# Loop para análise e visualização de cada atributo
for feature in features:
   X = df[[feature]]
   y = df['sales']

   # Adiciona o intercepto
   X_const = sm.add_constant(X)

   # Regressão linear
   model = sm.OLS(y, X_const).fit()

   # Predições
   y_pred = model.predict(X_const)
```

```
    # Gráfico
    plt.figure(figsize=(7, 5))
    sns.scatterplot(x=df[feature], y=y, label='Dados reais')
        plt.plot(df[feature], y_pred, color='red', label='Linha de
regressão')
        plt.title(f'Regressão Linear: {feature} vs Sales\nR² =
{model.rsquared:.3f}')
    plt.xlabel(feature)
    plt.ylabel('Sales')
    plt.legend()
    plt.tight_layout()
    plt.show()

    # Mostrar sumário
    print(f"\nResumo da Regressão para {feature}:\n")
    print(model.summary())
```

b- Utilizando os métodos corr (https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.corr.html#pandas.DataFrame.corr) e background_gradient (https://pandas.pydata.org/docs/reference/api/pandas.io.formats.style.Styler.background_gradient.html) do Pandas para gerar a matriz de correlação (Pearson e Spearman) entre os atributos do dataset.

Execute o código e faça uma análise dos resultados de acordo com o que foi discutido na última aula.
Compare o resultado obtido com os métodos pearsonr e spearmanr do pacote stats do Scipy. Neste último caso, apresente também o p-valor obtido.

```
import pandas as pd
import numpy as np
from scipy.stats import pearsonr, spearmanr

# Carregar o dataset (substitua pelo caminho real se necessário)
url =
"https://raw.githubusercontent.com/amankharwal/Website-data/master/advertising.csv"
df = pd.read_csv(url)

# --- Parte 1: Matriz de correlação com Pandas (Pearson e Spearman) ---
print("\n--- Matriz de Correlação (Pearson) ---")
corr_pearson = df.corr(method='pearson')
```

```
display(corr_pearson.style.background_gradient(cmap='coolwarm',
axis=None))

print("\n--- Matriz de Correlação (Spearman) ---")
corr_spearman = df.corr(method='spearman')
display(corr_spearman.style.background_gradient(cmap='coolwarm',
axis=None))

# --- Parte 2: Correlações individuais com Scipy (com p-valor) ---
print("\n--- Comparação com Scipy (Pearsonr e Spearmanr) ---")
attributes = df.columns[:-1]  # Exclui 'Sales' (target)
target = 'Sales'

for attr in attributes:
   # Pearson (r e p-valor)
   pearson_r, pearson_p = pearsonr(df[attr], df[target])
   # Spearman (r e p-valor)
   spearman_r, spearman_p = spearmanr(df[attr], df[target])

   print(f"\n**{attr} vs {target}**")
   print(f"Pearson:  r = {pearson_r:.3f}, p-valor = {pearson_p:.3e}")
   print(f"Spearman: r = {spearman_r:.3f}, p-valor = {spearman_p:.3e}")
```

c- Usando o método pairplot (https://seaborn.pydata.org/generated/seaborn.pairplot.html) do Seaborn para plotar a relação dos atributos dois a dois.
Pesquise sobre o que as saídas do pairplot trazem para análise. Execute o código e faça uma descrição dos resultados gráficos obtidos.

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Carregar o dataset
url =
"https://raw.githubusercontent.com/amankharwal/Website-data/master/advertising.csv"
df = pd.read_csv(url)

# Configurar o estilo do Seaborn
sns.set(style="ticks")

# Criar o pairplot
pairplot = sns.pairplot(
```

```
    data=df,
    vars=['TV', 'Radio', 'Newspaper', 'Sales'],  # Seleciona as colunas
    diag_kind='hist',  # Tipo de gráfico na diagonal (hist ou kde)
    plot_kws={'alpha': 0.6, 's': 30, 'edgecolor': 'k'},  # Ajustes dos
pontos
    height=2.5  # Altura de cada subplot
)
# Adicionar título
pairplot.fig.suptitle("Pairplot do Dataset Advertising", y=1.02)
# Ajustar layout e mostrar
plt.tight_layout()
plt.show()
```

d- Aplicando a regressão linear implementada a todas as combinações possíveis de atributos de entrada x atributo de saída. Lembre de gerar RSS, RSE e SE em cada um dos casos.

Execute o código e faça uma análise dos resultados de acordo com o que foi discutido na última aula.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
import statsmodels.api as sm

# Carregar o dataset
url =
"https://raw.githubusercontent.com/amankharwal/Website-data/master/adve
rtising.csv"
df = pd.read_csv(url)
X = df[['TV', 'Radio', 'Newspaper']]
y = df['Sales']

# Dicionário para armazenar resultados
test_results = {}

for i in range(10):
    # Dividir em treino (80%) e teste (20%) com random_state=i
    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=i)

    # --- Seleção dos 2 melhores atributos no treino (como antes) ---
    corr_with_sales =
X_train.corrwith(y_train).abs().sort_values(ascending=False)
```

```
    corr_matrix = X_train.corr().abs()
    selected_features = []
    for feature in corr_with_sales.index:
        if len(selected_features) < 2:
            if len(selected_features) == 0 or
all(corr_matrix.loc[feature, selected] < 0.7 for selected in
selected_features):
                selected_features.append(feature)

    # --- Treinar o modelo nos dados de treino ---
    X_train_selected = sm.add_constant(X_train[selected_features])
    model = sm.OLS(y_train, X_train_selected).fit()

    # --- Aplicar ao conjunto de teste e calcular RSS/RSE ---
    X_test_selected = sm.add_constant(X_test[selected_features])
    y_pred = model.predict(X_test_selected)
    residuals = y_test - y_pred
    rss = np.sum(residuals**2)
    rse = np.sqrt(rss / (len(y_test) - len(selected_features) - 1))  #
Ajustado para graus de liberdade

    # Armazenar resultados
    test_results[f"Split_{i+1}"] = {
        'Features': selected_features,
        'RSS_test': rss,
        'RSE_test': rse,
        'R2_test': model.rsquared_adj  # R² ajustado para comparação
    }

# Exibir resultados
print("Resultados nos Conjuntos de Teste:")
for split, data in test_results.items():
    print(f"\n{split}: Features = {data['Features']}")
    print(f"  RSS: {data['RSS_test']:.2f}")
    print(f"  RSE: {data['RSE_test']:.2f}")
    print(f"  R² ajustado: {data['R2_test']:.4f}")
    print("=" * 50)
```

e- Aplicando a regressão linear do scikit-learn (https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html) e compare com os resultados obtidos em a.

Execute o código e faça uma análise dos resultados comparando com os obtidos pela letra a.

```
import pandas as pd
import statsmodels.api as sm
import seaborn as sns
import matplotlib.pyplot as plt

# Carrega o dataset
url =
'https://raw.githubusercontent.com/selva86/datasets/master/Advertising.
csv'
df = pd.read_csv(url, index_col=0)

# Definindo variáveis independentes e dependente
X = df[['TV', 'radio', 'newspaper']]
y = df['sales']

# Adiciona intercepto
X_const = sm.add_constant(X)

# Ajuste do modelo
model = sm.OLS(y, X_const).fit()

# Predições
y_pred = model.predict(X_const)
# Exibir sumário da regressão
print(model.summary())
# Plotando os resíduos
residuals = y - y_pred
sns.residplot(x=y_pred, y=residuals, lowess=True, line_kws={'color':
'red'})
plt.xlabel('Valores Previstos')
plt.ylabel('Resíduos')
plt.title('Resíduos vs Valores Previstos')
plt.axhline(0, color='black', linestyle='--', lw=1)
plt.tight_layout()
plt.show()
```