untitled124

August 25, 2024

[1]:
```
# importando bibliotecas
import pandas as pd
import numpy as np
import sklearn.linear_model as linear_model
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.graph_objects as go
import plotly.express as px
```

[3]:
```
# Importando a dataset
data = pd.read_csv('/content/MKT.csv')
```

[4]:
```
# Vendo os 10 primeiros registros
data.head(10)
```

[4]:

	youtube	facebook	newspaper	sales
0	84.72	19.20	48.96	12.60
1	351.48	33.96	51.84	25.68
2	135.48	20.88	46.32	14.28
3	116.64	1.80	36.00	11.52
4	318.72	24.00	0.36	20.88
5	114.84	1.68	8.88	11.40
6	348.84	4.92	10.20	15.36
7	320.28	52.56	6.00	30.48
8	89.64	59.28	54.84	17.64
9	51.72	32.04	42.12	12.12

[5]:
```
# Exibindo informações gerais
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 171 entries, 0 to 170
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   youtube    171 non-null    float64
 1   facebook   171 non-null    float64
 2   newspaper  171 non-null    float64
```

```
 3   sales      171 non-null    float64
dtypes: float64(4)
memory usage: 5.5 KB
```

[6]:
```
# Verificando dados nulos
data.isnull()
```

[6]:
```
     youtube  facebook  newspaper  sales
0      False     False      False  False
1      False     False      False  False
2      False     False      False  False
3      False     False      False  False
4      False     False      False  False
..       …         …        …     …
166    False     False      False  False
167    False     False      False  False
168    False     False      False  False
169    False     False      False  False
170    False     False      False  False

[171 rows x 4 columns]
```

[7]:
```
# Verificando se existem valores duplicados
data.duplicated()
```

[7]:
```
0      False
1      False
2      False
3      False
4      False
       …
166    False
167    False
168    False
169    False
170    False
Length: 171, dtype: bool
```

[8]:
```
# Verificando os tipos de dados
data.dtypes
```

[8]:
```
youtube      float64
facebook     float64
newspaper    float64
sales        float64
dtype: object
```

[9]:
```
# Descrição dos dados
data.describe()
```

[9]:

	youtube	facebook	newspaper	sales
count	171.000000	171.000000	171.000000	171.000000
mean	178.021053	27.671579	35.240000	16.922807
std	102.449597	17.913532	24.902918	6.314608
min	0.840000	0.000000	0.360000	1.920000
25%	91.080000	11.700000	13.740000	12.540000
50%	179.760000	26.760000	31.080000	15.480000
75%	262.980000	43.680000	50.880000	20.820000
max	355.680000	59.520000	121.080000	32.400000

ANÁLISE DESCRITIVA

Total de linhas: 171 registros.

Total de colunas: 4 colunas (Youtube, Facebook, Newspaper, Sales).

Colunas Youtube, Facebook e Neswpaper: Valores dos investimentos mensais em cada plataforma.

Coluna Sales: Valor das vendas mensais.

Tipo dos dados: float.

Dados nulos: sem dados nulos.

Dados duplicados: sem dados duplicados.

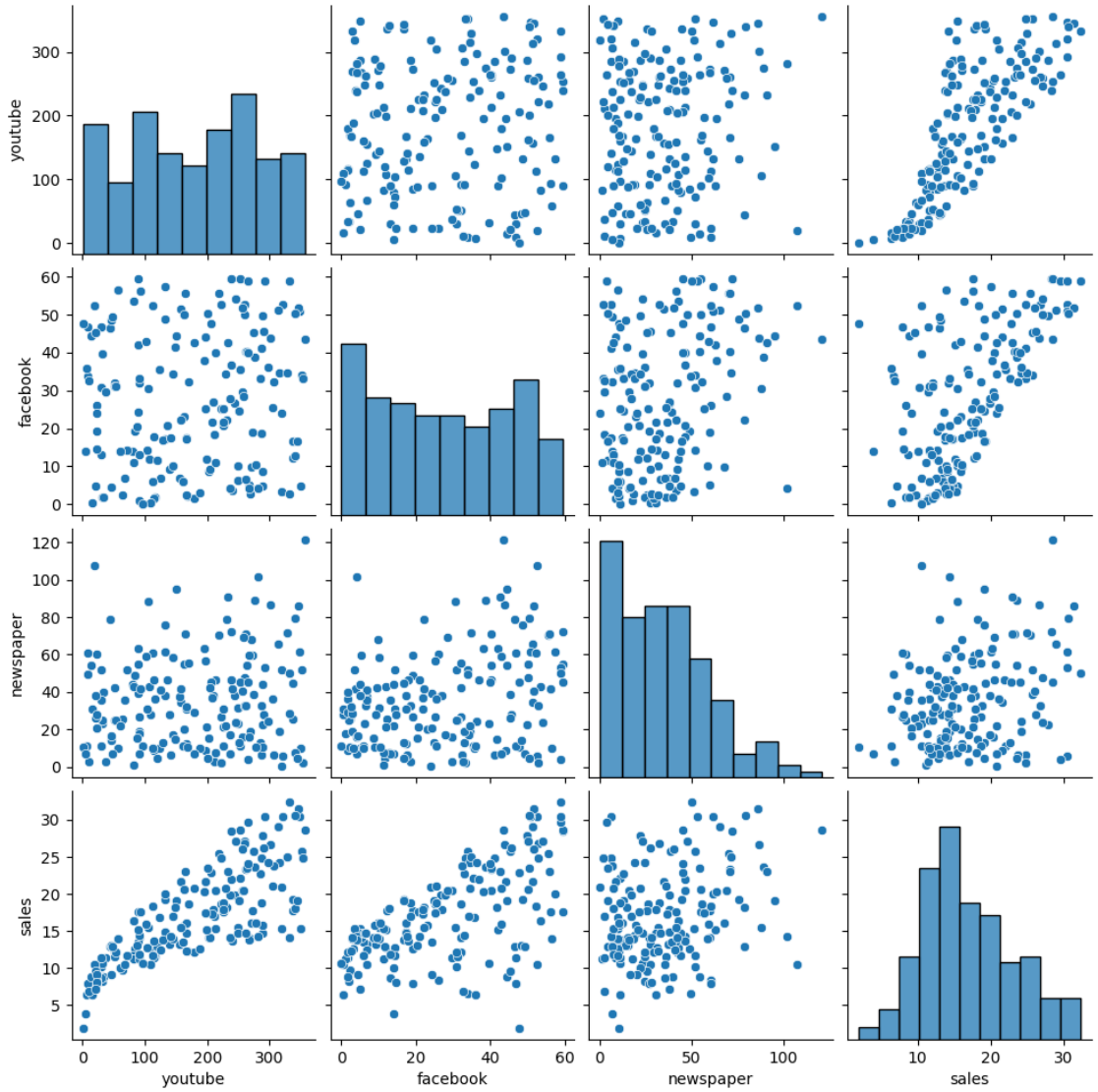
ANÁLISE EXPLORATÓRIA

Analisando os dados com um gráfico de correleção vemos as seguintes informações:

Os maiores investimentos estão concentrados na plataforma do Youtube, seguido por Newspaper e por ultimo o Facebook. Quanto maior o investimento nas plataformas como o Youtube e o Facebook, maior a probabilidade de vendas. No caso do Newspaper, o valor do invesimento comparado as vendas não existe muita correlação, visto que com valores baixos de investimento obteve-se um valor significativo de vendas em determinados meses. A média de vendas fica na casa R$16.000.

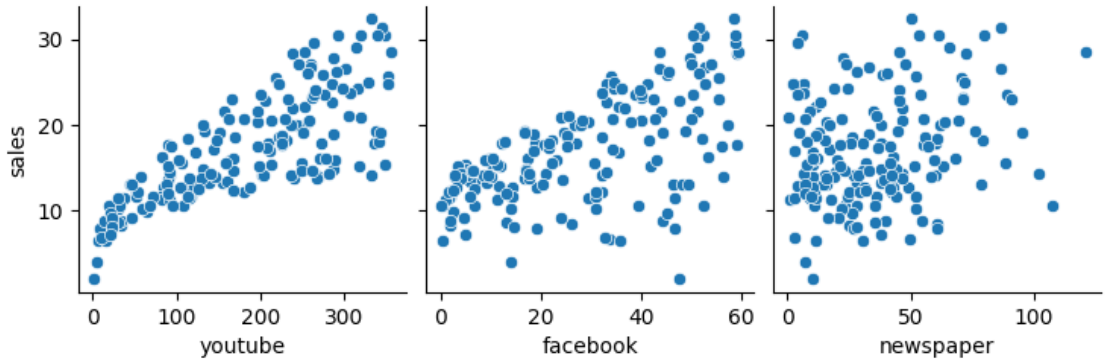
[10]:
```
# Gráfico de correlações entre os dados
sns.pairplot(data)
```

[10]:
```
<seaborn.axisgrid.PairGrid at 0x7e9ffec55660>
```



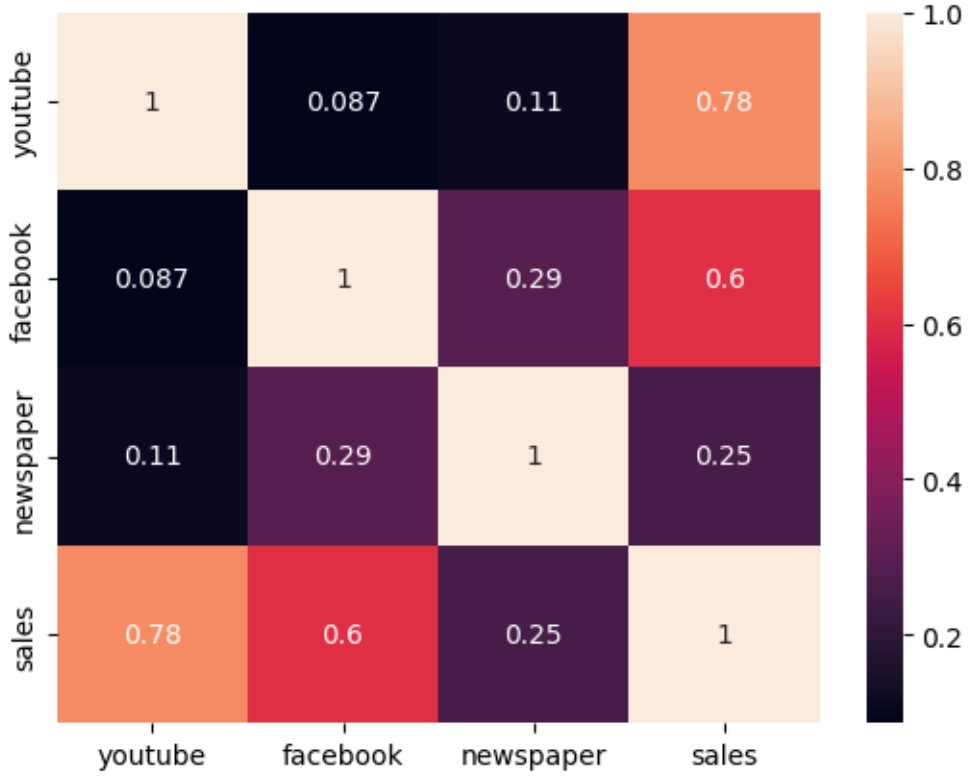
[11]:
```
# Correlação de plataforma vs sales
sns.pairplot(data,x_vars=['youtube','facebook','newspaper'], y_vars=['sales'])
```

[11]: <seaborn.axisgrid.PairGrid at 0x7e9ffb9e73a0>



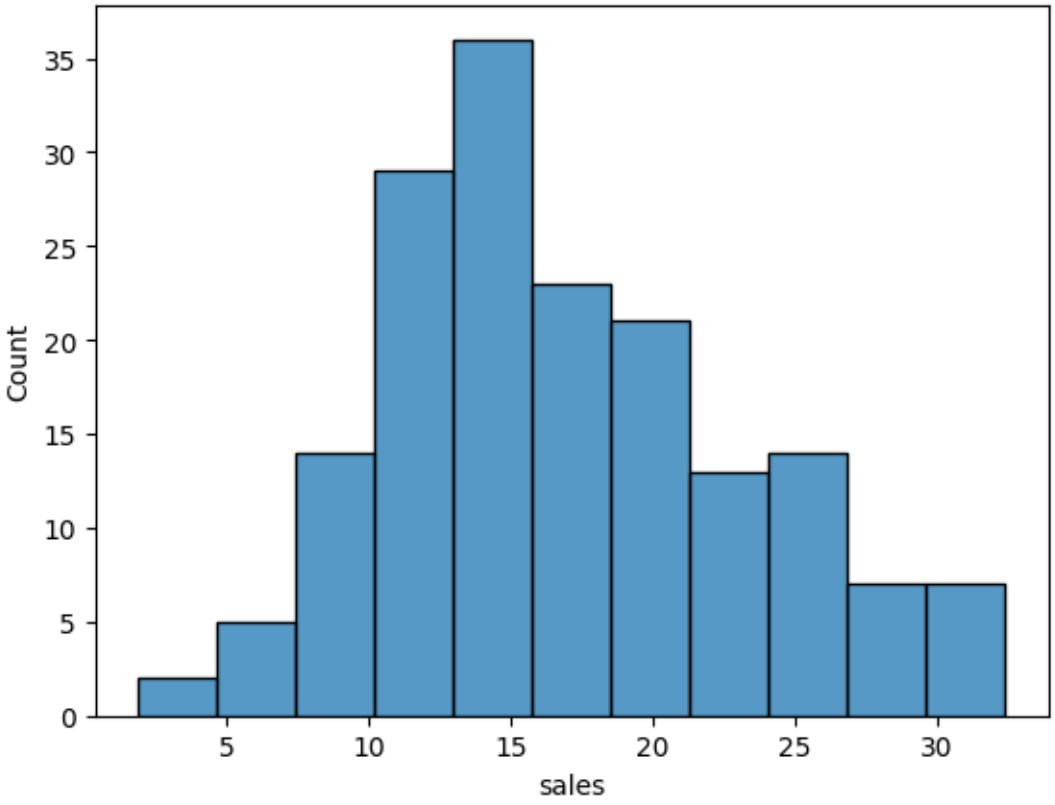
[12]:
```
# Gráfico de mapa de calor
sns.heatmap(data.corr(), annot=True)
```

[12]: <Axes: >



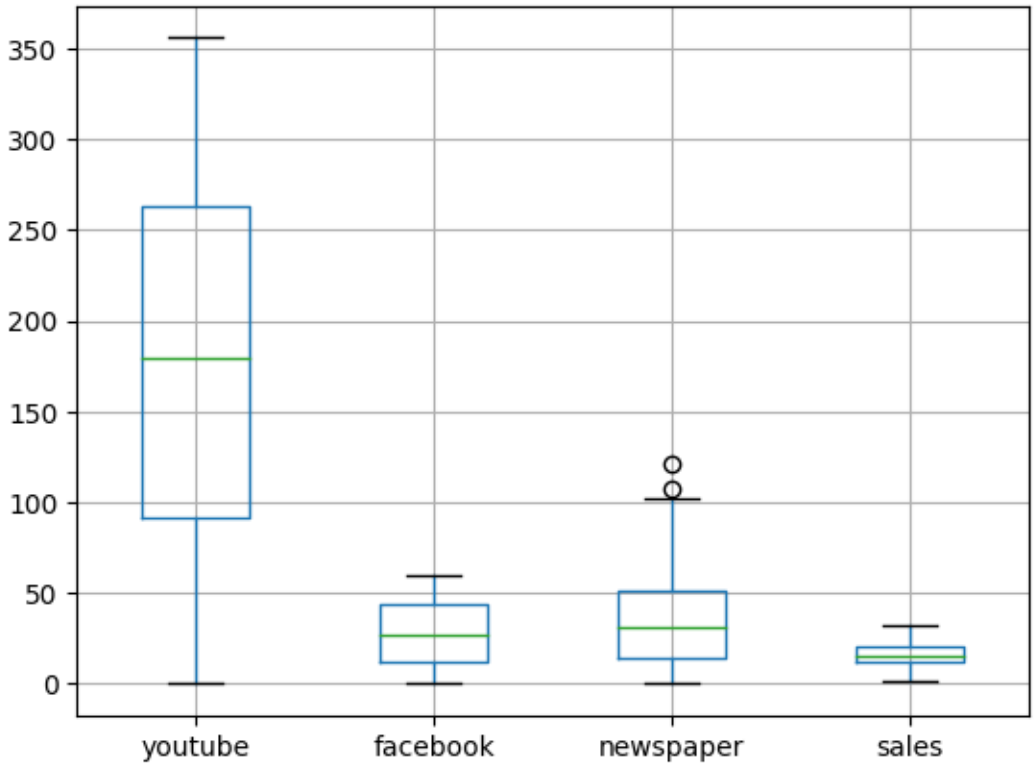
[13]:
```
# Histograma da variável 'sales'
sns.histplot(data['sales'])
```

[13]: <Axes: xlabel='sales', ylabel='Count'>



[14]:
```
# Análise de distribuição dos valores
data.boxplot()
```

[14]: <Axes: >



MODELAGEM

Na etapa de modelagem, iremos criar um modelo simples de regressão linear que permita a previsão solicitada pela empresa.

Criar um modelo de Regressão Linear do SKLEARN. Treinar e testar o respectivo modelo. DESIGN DO MODELO

Fazer a separação de train/test do dataset com 20% de massa para teste via o método do SKLEARN. Usaremos o modelo 'train_test_split', também do SKLEARN. MÉTRICAS PARA AVALIAÇÃO DO MODELO

Realizar o cálculo do coeficiente de determinação (r^2). Com o resultado poder predizer o valor das vendas de acordo com o investimento em cada plataforma.

[15]:
```
# Modelagem, importando a a biblioteca de treino
from sklearn.model_selection import train_test_split

# Definindo minha variável data e variável target
X = data[['youtube','facebook','newspaper']]
Y = data[['sales']]
```

[16]:
```
# Definindo variáveis train/test
```

```
X_train, X_test, Y_train, Y_test= train_test_split(X,Y, train_size = 0.8,␣
↪test_size = 0.2, random_state = 42)
```

[17]:
```
print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)
print(Y_test.shape)
```

```
(136, 3)
(35, 3)
(136, 1)
(35, 1)
```

[18]:
```
# Importando o modelo de Regressão Linear do SKLEARN
from sklearn.linear_model import LinearRegression
```

[19]:
```
# Definindo a variável de RL e treinando
rl = LinearRegression()
rl = LinearRegression().fit(X_train, Y_train)
```

[20]:
```
# Verificando o modelo
rl
```

[20]:
```
LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook. On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

[21]:
```
# Variável de predição do treino
y_pred = rl.predict(X_test)
```

[22]:
```
# Importando o coeficiente de determinação(r^2 ) e definindo-o.
from sklearn.metrics import r2_score
r2 = r2_score(Y_test,y_pred)
```

[23]:
```
# Prevendo o resultado em %
print(f'O valor do coeficiente de determinação é de: {round(r2*100,2)} %')
```

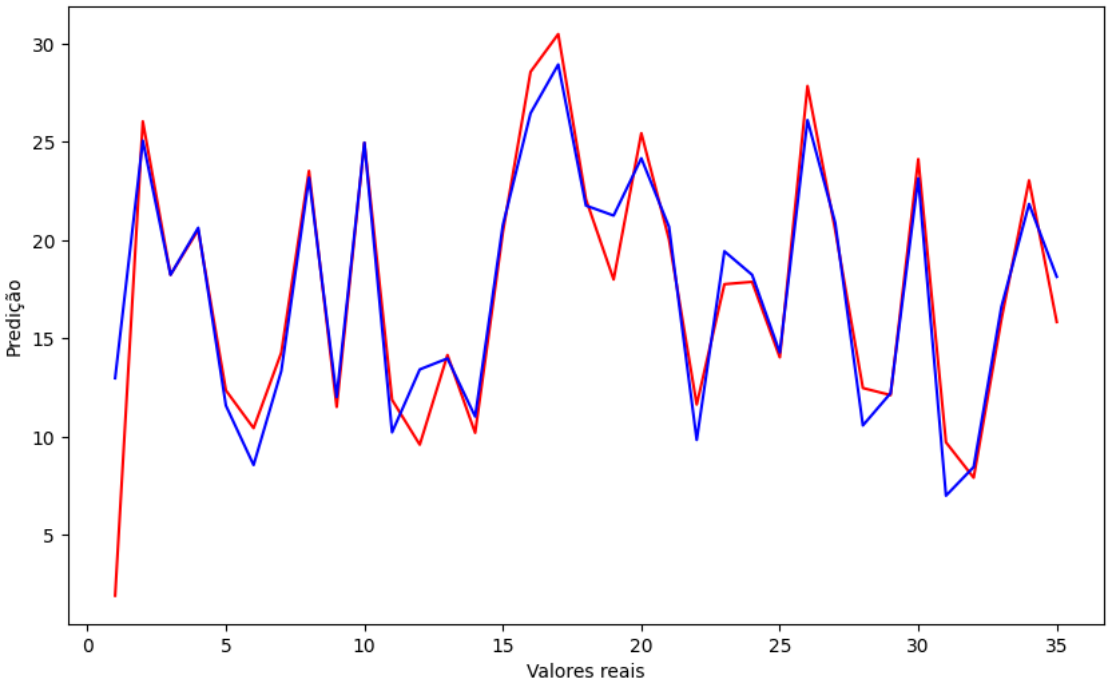
```
O valor do coeficiente de determinação é de: 87.29 %
```

[24]:
```
# Plotando gráfico com o valor do R²
c = [i for i in range(1,36,1)]
fig = plt.figure(figsize=(10,6))

plt.plot(c,Y_test, color='red')
plt.plot(c,y_pred, color='blue')
```

```
plt.xlabel('Valores reais')
plt.ylabel('Predição')
```

[24]: `Text(0, 0.5, 'Predição')`



CONCLUSÃO

Com a respectiva modelagem dos dados e treino da nossa base conseguimos chegar ao nosso objetibo e determinar que:

Conseguimos um valor de coeficiente de determinação, o r^2, de 87,29%. Comparando o gráfico de Valores reais vs Predição, vimos que nossa predição de valores é bastante confiavel para estimar possiveis retornos de vendas em calculos posteriores que podem ser gerados a partir de determinados investimentos.