Entrega 2 | Álgebra Linear, Vetores e Geometria Analítica

Tema: Construção de um Modelo de Regressão Linear

Introdução

Para a continuação do projeto optamos por seguir com outra base de dados. Pois essa, além de conter mais dados (1200 dados, em comparação com a da entrega anterior, que possuía 200), possui os campos de NPS (Net Promoter Score) e Rating, que serão o foco da análise.

O primeiro passo para a construção do Modelo de Regressão Linear foi a inserção dos arquivos no ambiente do colab. No caso embora nosso foco seja na tabela de pedidos optei por incluir também as outras tabelas para caso seja necessário futuramente.

```
import pandas as pd
import json
```

Confirmando se os arquivos estão funcionando corretamente

```
dfCustomers = pd.read_csv('customers.csv')
dfOrders = pd.read_csv('orders.csv')
dfMenu_Items = pd.read_csv('menu_items.csv')
dfRestaurants = pd.read_csv('restaurants.csv')
dfOrder_Items = pd.read_csv('order_items.csv')
```

Agora verificando se os arquivos foram incluídos corretamente

```
print("Primeiras linhas da tabela pedidos")
dfOrders.head(10)
```

```
Primeiras linhas da tabela pedidos
```

	id	restaurant_id	customer_id	order_value	delivery_time_minutes	rating	nps_score	created_at
0	8b7a4852-a354-47ed-a610-e4c2e020b6ad	393b0488-a823-4ce2-ae97-98fb945c18ce	e1fd11b6-db7f-4a20-b35f-eaeff64d356d	43.60	24	5	87	2025-10-01 20:51:16.1621+00
1	44dd0f03-1116-47b0-b513-319b0223b5b1	baa12475-6c55-459a-8e94-00990d918b00	427bb311-3e9c-46b0-bc35-a2abaeb777b8	99.66	37	4	51	2025-09-26 01:26:58.181927+00
2	aa464656-3f09-46fa-898f-dd208ae90272	f7ac584e-097e-4b23-b8d7-131800400f4c	f191d7b3-31fc-4b16-90a0-90b9270ee51f	760.22	32	4	38	2025-08-03 20:18:03.192971+00
3	e012c512-46b4-43fa-b053-1cb2e6cc1325	f6f9d5c3-4286-42ae-aebb-388aa6b5bb9f	94276f94-5b83-4d47-bd70-a3dac590c832	265.89	44	2	-95	2025-06-17 22:39:37.210549+00
4	3d7a9417-ca7d-4657-97fd-009e9e4340eb	b639ee7f-52bb-4e7c-b4f7-395de58cfc6b	c1b1854f-b150-40ce-88e5-5c44cab55ddb	519.74	39	1	-12	2025-05-17 14:50:12.221616+00
5	fbb2d3b7-cdea-4f93-b46a-b8e8364c7f2c	b76cc620-6b3d-4e05-8342-3ffe9b5c5d8e	5912c207-b544-4ca7-905e-26139080a3f6	57.78	10	5	92	2025-06-16 18:12:54.232918+00
6	5ad2c67c-03f3-472b-adb0-f0f8d6d87a58	ec3b3163-5344-4bfb-944d-36f8495eda50	fdf3b28b-ed1d-4fca-8a14-329d3890f53f	375.70	19	5	76	2025-08-24 02:57:54.240759+00
7	7ca3ccd0-e91a-4e08-b15a-32af5d378751	4ea0a6d1-b20a-4751-bc85-0f0758c4838d	18444b6e-c4ff-4e45-b6a0-827cf968c1cb	648.91	19	3	25	2025-05-21 18:56:10.250353+00
8	bbd32694-c397-420a-acca-75f68b726559	b639ee7f-52bb-4e7c-b4f7-395de58cfc6b	ec0ecb79-4fa2-49af-9816-7647f4f3c059	402.83	35	1	-45	2025-06-01 14:22:03.26274+00
9	5b724676-9200-4f2a-b23d-ebabc245e35d	ca1b4847-ddde-4120-983c-cc1ca00b400f	43ed865d-bcdb-4a1d-80ea-b3241c4c79b1	298.43	50	5	83	2025-10-24 00:47:20.278609+00

Campos Utilizados

Durante a construção do modelo utilizamos Cinco campos. Sendo eles:

order_value – Valor total do pedido.

delivery_time_minutes – Tempo total da entrega do pedido (desde a preparação até a entrega).

created_at – Data e horário em que o pedido foi feito.

rating – Nota de 1 a 5 a respeito da experiência do cliente.

nps_score – o NPS é uma forma de mensurar a fidelidade e experiência do cliente, incluindo questões como “você recomendaria o produto para um amigo?”.

Separando o campo de data

Para dar início a análise era necessário separar o campo created_at em dois, sendo um para a data em que o pedido foi efetuado e outro para o horário em que ele foi pedido.

```
from datetime import datetime


dfOrders['created_at'] = (
    pd.to_datetime(dfOrders['created_at'])
    .dt.tz_localize(None)
    .dt.floor('s')
)

dfOrders['Data'] = dfOrders['created_at'].dt.date
dfOrders['Hora'] = dfOrders['created_at'].dt.time

dfOrders[["Data", "Hora"]].head()
```

	Data	Hora
0	2025-10-01	20:51:16
1	2025-09-26	01:26:58
2	2025-08-03	20:18:03
3	2025-06-17	22:39:37
4	2025-05-17	14:50:12

Criação de novos campos

Em seguida utilizamos o campo Data para descobrir o dia da semana do pedido, sendo 0 Domingo e 6 Sábado.

```
dfOrders['Data'] = pd.to_datetime(dfOrders['Data'])
dfOrders['dia_da_semana'] = dfOrders['Data'].dt.day_of_week
```

```
datetime64[ns]
```

```
dfOrders[['Data', 'dia_da_semana']].head()
```

	Data	dia_da_semana
0	2025-10-01	2
1	2025-09-26	4
2	2025-08-03	6
3	2025-06-17	1
4	2025-05-17	5

Além disso, criamos uma outra coluna com o objetivo de informar a quantidade de minutos a partir da meia noite (por exemplo, se o pedido foi feito às 00:34, o valor será 34.), que será essencial para a regressão linear.

```
dfOrders['Hora'] = pd.to_datetime(dfOrders['Hora'], format='%H:%M:%S')

# nova coluna com minutos desde meia-noite
dfOrders['hora_minutos'] = dfOrders['Hora'].dt.hour * 60 + dfOrders['Hora'].dt.minute

print(dfOrders[['Hora', 'hora_minutos']].head())
```

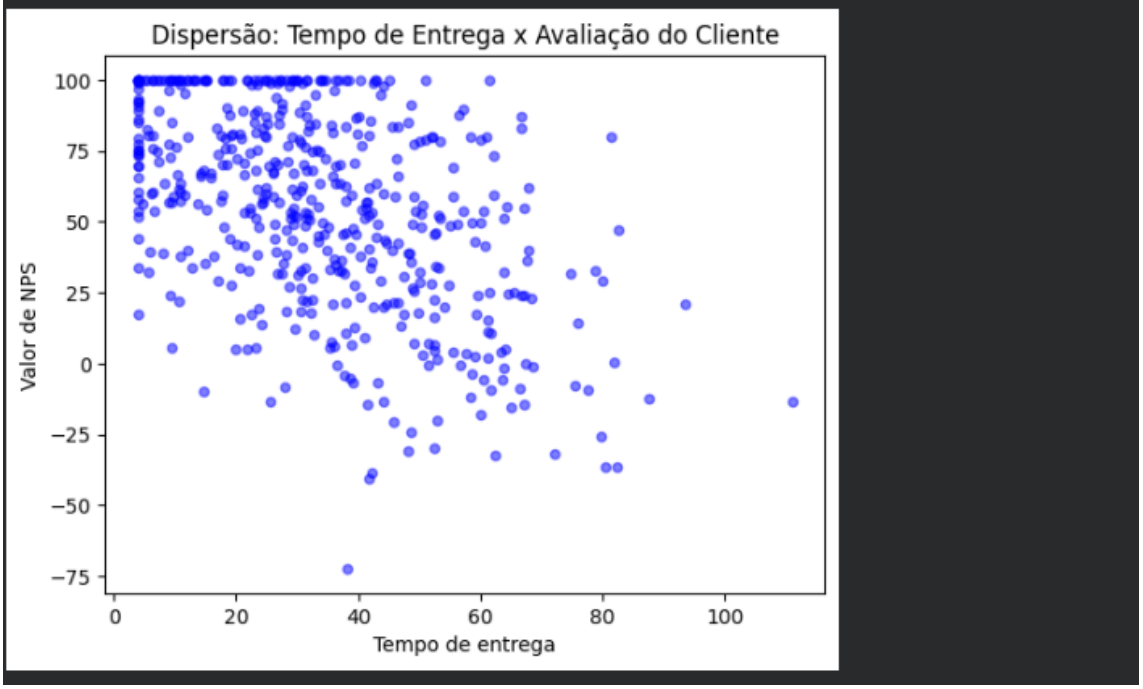
```
                 Hora  hora_minutos
0 1900-01-01 20:51:16          1251
1 1900-01-01 01:26:58            86
2 1900-01-01 20:18:03          1218
3 1900-01-01 22:39:37          1359
4 1900-01-01 14:50:12           890
```

Plotagem de gráficos

Agora, com os campos devidamente formados, temos tudo que precisamos para fazer gráficos e análises.

Tempo de Entrega x Avaliação do Cliente

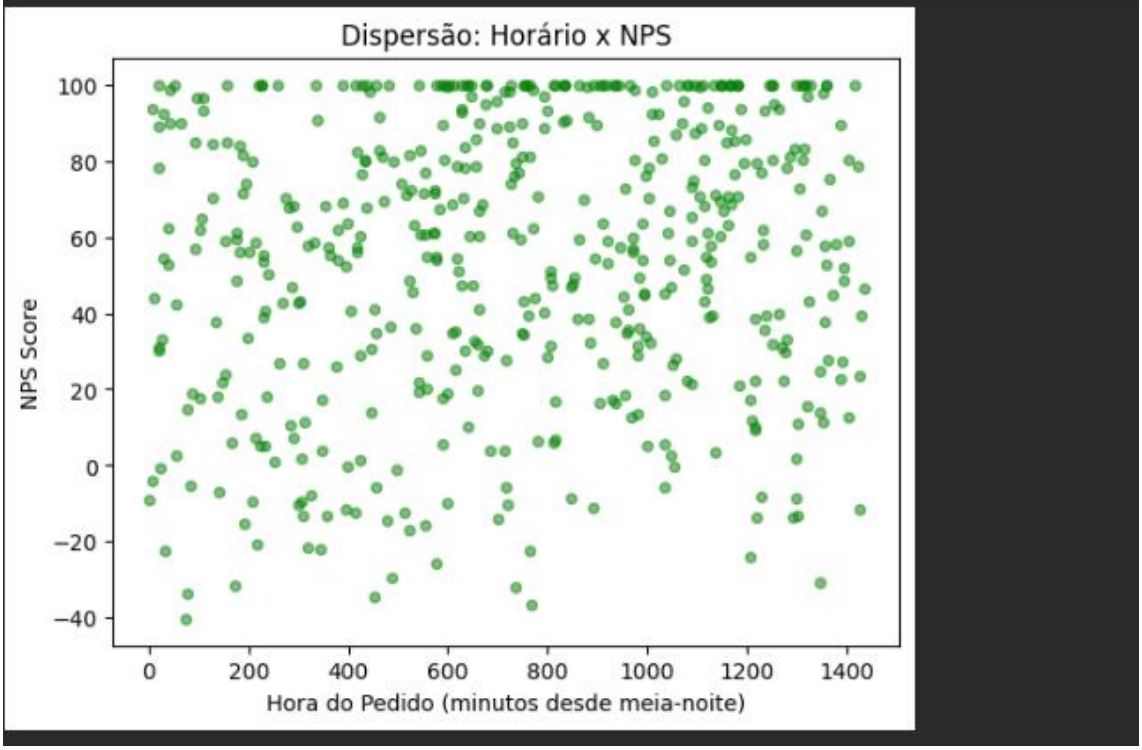
```
import matplotlib.pyplot as plt
dfOrders.sample(500).plot.scatter(x='delivery_time_minutes', y='nps_score', color='blue', alpha=0.5)
plt.xlabel('Tempo de entrega')
plt.ylabel('Valor de NPS')
plt.title('Dispersão: Tempo de Entrega x Avaliação do Cliente')
plt.show()
```



Aqui podemos perceber uma correlação inversa entre NPS e Tempo de Entrega. Um exemplo disso é que quando o tempo de entrega é baixo há uma concentração de NPS próximos de 100, e quanto maior o tempo de entrega ocorre uma dispersão dos valores.

Horário do Pedido x NPS

```
dfOrders.sample(500).plot.scatter(x='hora_minutos', y='nps_score', color='green', alpha=0.5)
plt.xlabel('Hora do Pedido (minutos desde meia-noite)')
plt.ylabel('NPS Score')
plt.title('Dispersão: Horário x NPS')
plt.show()
```



Vale destacar que, os valores numéricos representam o horário do dia, desta forma:

0 - **00:00** | 200 - **3:20**

400 - **6:40** | 600 - **10:00**

800 - **13:20** | 1000 – **16:40**

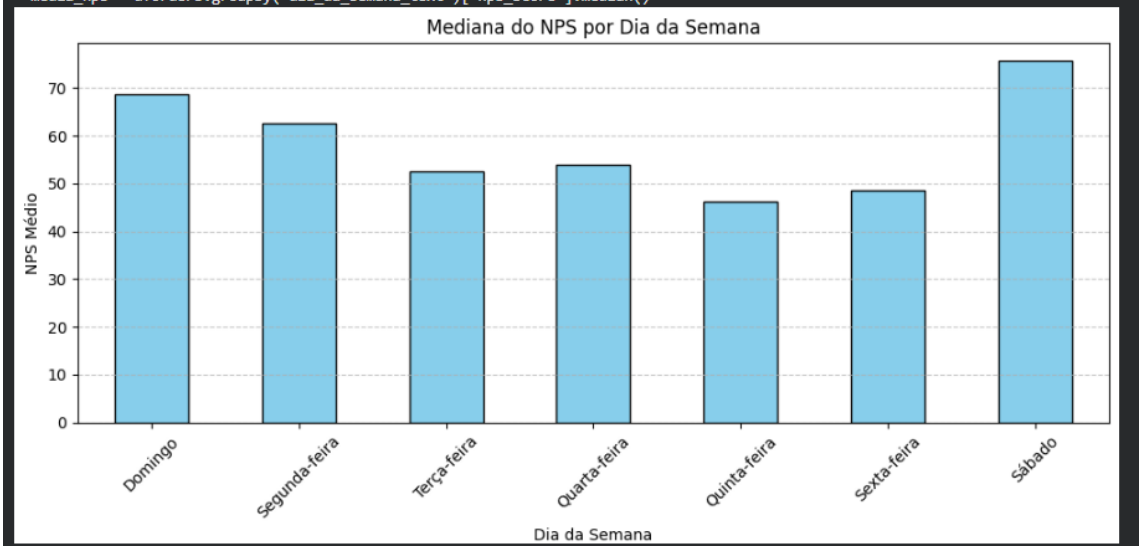
1200 - **20:00** | 1400 - **23:20**

Assim é possível notar uma grande concentração de pedidos nos períodos da tarde/noite, e uma certa diminuição no período da madrugada e manhã (entre 200 a 600), porém a relação entre horário do pedido e NPS é bem uniforme, sem grandes variações entre horários.

NPS por dia da semana

```
plt.title('Mediana do NPS por Dia da Semana')
plt.xlabel('Dia da Semana')
plt.ylabel('NPS Médio')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
print(dfOrders.columns)
```

```
/tmp/ipython-input-2843988584.py:18: FutureWarning: The default of observed=False is deprecated and will be changed to True in a fu
  media_nps = dfOrders.groupby('dia_da_semana_text')['nps_score'].median()
```



Como podemos observar, há uma alta na mediana do NPS durante o final de semana, que também se mantém alta na segunda feira e possui certa queda nos dias seguintes.

Desenvolvendo o Modelo de Regressão Linear

Após analisar as correlações entre campos dos nossos dados chegou o momento de criar um modelo de inteligência artificial com o objetivo de criar uma previsão tanto do NPS quanto do Tempo de Entrega baseado em 3 métricas, sendo elas: Valor do Pedido, Dia da Semana e Horário do Pedido.

```
import numpy as np
from scipy.linalg import lstsq

for col in ['nps_score', 'delivery_time_minutes', 'order_value', 'hora_minutos']:
    dfOrders[col] = pd.to_numeric(dfOrders[col], errors='coerce')

df_model = dfOrders.dropna(subset=['nps_score', 'delivery_time_minutes', 'order_value', 'hora_minutos', 'dia_da_semana']).copy()


X_categorical = pd.get_dummies(df_model['dia_da_semana'], prefix='dia', drop_first=True)


X_continuous = df_model[['delivery_time_minutes', 'order_value', 'hora_minutos']]


X_base = pd.concat([X_continuous, X_categorical], axis=1)


X_base_values_float = X_base.values.astype(np.float64)


X = np.hstack([np.ones((X_base.shape[0], 1)), X_base_values_float])


Y = df_model['nps_score'].values.astype(np.float64).reshape(-1, 1)

print(f"Dimensão da Matriz X: {X.shape}")
print(f"Dimensão do Vetor Y: {Y.shape}")
print(f"Tipo de dado de X: {X.dtype}")
print(f"Tipo de dado de Y: {Y.dtype}")
df_model.to_csv('df_model.csv', index=False)


beta_coefficients, residuals, rank, s = lstsq(X, Y)
```

Neste primeiro trecho realizamos a limpeza dos dados, a construção da matriz de features e a resolução da regressão linear com o objetivo de prever o NPS. Começamos convertendo as colunas em campos numéricos e retirando os valores nulos, para que possam ser utilizados no modelo. Após isso, combinamos as variáveis contínuas (valores que podem ser variados) e categóricas (valores classificados em grupos, por exemplo o campo *dia_da_semana*, que pode ter apenas os valores de 0 a 6) na variável `X_base`. Por fim, adicionamos o intercepto, criamos o vetor resposta `Y` com os valores reais de NPS e aplicamos o método de mínimos quadrados para obter os coeficientes do modelo preditivo.

```python
Y_B = df_model['delivery_time_minutes'].values.reshape(-1, 1)

X_B_continuous = df_model[['order_value', 'hora_minutos']]

X_B_categorical = pd.get_dummies(df_model['dia_da_semana'], prefix='dia', drop_first=True)

X_B_base = pd.concat([X_B_continuous, X_B_categorical], axis=1)

X_B = np.hstack([np.ones((X_B_base.shape[0], 1)), X_B_base.values])

print(f"Dimensão da Matriz X_B: {X_B.shape}")
print(f"Dimensão do Vetor Y_B: {Y_B.shape}")
```

```
Dimensão da Matriz X_B: (1000, 9)
Dimensão do Vetor Y_B: (1000, 1)
```

Neste trecho preparamos um segundo modelo, desta vez com o objetivo de prever o tempo de entrega dos pedidos. Utilizamos como variáveis explicativas o valor do pedido, o horário em que foi feito e o dia da semana. Assim como no modelo anterior, as variáveis categóricas passam por um processo de *One-Hot Encoding* e as variáveis contínuas são mantidas em sua forma numérica. Ao final, unimos todas essas informações em uma matriz de features (`X_B`) e preparamos o vetor resposta (`Y_B`) contendo os tempos de entrega observados.

```python
Y_B = df_model['delivery_time_minutes'].values.astype(np.float64).reshape(-1, 1)

X_B_continuous = df_model[['order_value', 'hora_minutos']]

X_B_categorical = pd.get_dummies(df_model['dia_da_semana'], prefix='dia', drop_first=True)

X_B_base = pd.concat([X_B_continuous, X_B_categorical], axis=1)

X_B_values_float = X_B_base.values.astype(np.float64)

X_B = np.hstack([np.ones((X_B_base.shape[0], 1)), X_B_values_float])

print(f"Dimensão da Matriz X_B: {X_B.shape}")
print(f"Dimensão do Vetor Y_B: {Y_B.shape}")

print(f"Tipo de dado de X_B: {X_B.dtype}")
print(f"Tipo de dado de Y_B: {Y_B.dtype}")

beta_B_coefficients, residuals_B, rank_B, s_B = lstsq(X_B, Y_B)
```

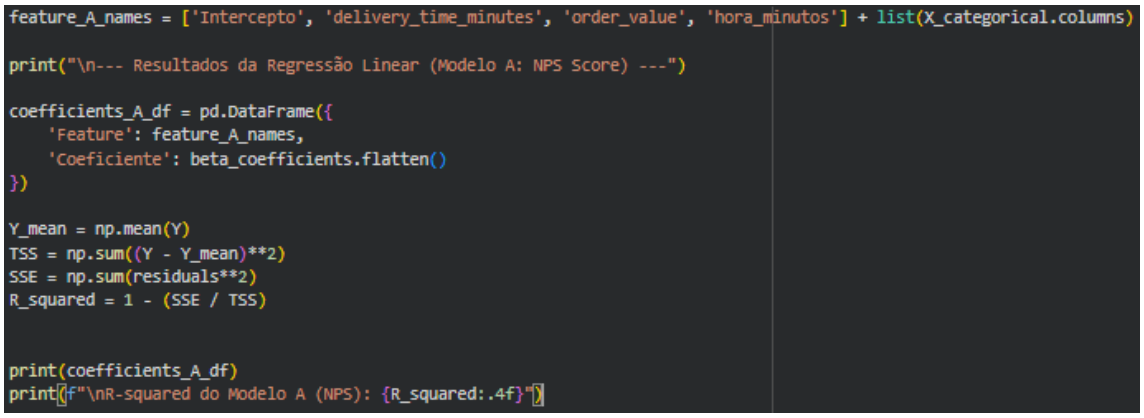
```
Dimensão da Matriz X_B: (1000, 9)
Dimensão do Vetor Y_B: (1000, 1)
Tipo de dado de X_B: float64
Tipo de dado de Y_B: float64
```

Neste trecho realizamos o ajuste final do Modelo B, garantindo que todos os dados estejam no formato numérico correto (`float64`) antes de aplicar novamente o método dos mínimos quadrados. Essa conversão é essencial para evitar erros de tipagem e permitir que o cálculo da regressão linear ocorra corretamente. Com isso, obtemos os coeficientes (`beta_B_coefficients`) que representam a influência de cada variável no tempo de entrega.

```python
feature_B_names = ['Intercepto'] + list(X_B_base.columns)

print("\n--- Resultados da Regressão Linear (Modelo B: Tempo de Entrega) ---")

coefficients_B_df = pd.DataFrame({
    'Feature': feature_B_names,
    'Coeficiente': beta_B_coefficients.flatten()
})


Y_B_mean = Y_B.mean()
TSS_B = np.sum((Y_B - Y_B_mean)**2)
SSE_B = np.sum(residuals_B**2)
R_squared_B = 1 - (SSE_B / TSS_B)


print(coefficients_B_df)
print(f"\nR-squared do Modelo B (Tempo de Entrega): {R_squared_B:.4f}")
```

```
--- Resultados da Regressão Linear (Modelo B: Tempo de Entrega) ---
        Feature  Coeficiente
0    Intercepto    24.263052
1   order_value     0.040062
2  hora_minutos    -0.009730
3         dia_1     8.455335
4         dia_2     9.718233
5         dia_3    12.232670
6         dia_4    22.441234
7         dia_5    23.686532
8         dia_6    -0.917720

R-squared do Modelo B (Tempo de Entrega): -184923.2241
```

Aqui avaliamos o desempenho do Modelo B. São exibidos os coeficientes estimados para cada variável e calculado o valor de R^2 (R-quadrado), que indica o quanto o modelo consegue explicar da variação observada no tempo de entrega. Quanto mais próximo de 1 o R^2 estiver, melhor o modelo representa os dados reais. Esse passo é fundamental para interpretar o peso de cada variável e entender a qualidade da previsão.



```
feature_A_names = ['Intercepto', 'delivery_time_minutes', 'order_value', 'hora_minutos'] + list(X_categorical.columns)

print("\n--- Resultados da Regressão Linear (Modelo A: NPS Score) ---")

coefficients_A_df = pd.DataFrame({
    'Feature': feature_A_names,
    'Coeficiente': beta_coefficients.flatten()
})

Y_mean = np.mean(Y)
TSS = np.sum((Y - Y_mean)**2)
SSE = np.sum(residuals**2)
R_squared = 1 - (SSE / TSS)


print(coefficients_A_df)
print(f"\nR-squared do Modelo A (NPS): {R_squared:.4f}")
```

Neste trecho avaliamos o Modelo A, responsável pela previsão do NPS. Assim como no modelo anterior, criamos uma tabela com os coeficientes de cada variável e calculamos o R^2 para medir a precisão do modelo. O objetivo aqui é entender como fatores como tempo de entrega, valor do pedido, horário e dia da semana influenciam diretamente o NPS, que representa o nível de satisfação do cliente.

```python
def prever_nps_e_tempo_entrega(valor_pedido, hora_minutos_pedido, dia_da_semana_num, beta_nps, beta_entrega):
    """
    Recebe inputs do usuário e retorna a previsão de Tempo de Entrega e NPS.

    Args:
        valor_pedido (float): order_value do pedido.
        hora_minutos_pedido (int): hora_minutos do pedido (total de minutos após 00:00).
        dia_da_semana_num (int): 0 (Dom) a 6 (Sáb).
        beta_nps (array): Coeficientes do Modelo A (NPS).
        beta_entrega (array): Coeficientes do Modelo B (Tempo de Entrega).
    """


    dia_dummies = np.zeros(6)
    if dia_da_semana_num > 0 and dia_da_semana_num <= 6:
        dia_dummies[dia_da_semana_num - 1] = 1
    X_B_pred = np.array([1, valor_pedido, hora_minutos_pedido] + list(dia_dummies))

    tempo_entrega_previsto = X_B_pred.dot(beta_entrega)[0]

    X_A_pred = np.array([1, tempo_entrega_previsto, valor_pedido, hora_minutos_pedido] + list(dia_dummies))

    nps_previsto = X_A_pred.dot(beta_nps)[0]

    return tempo_entrega_previsto, nps_previsto
```

A função `prever_nps_e_tempo_entrega` calcula previsões de **tempo de entrega** e **NPS (Net Promoter Score)** com base em informações de um pedido e em coeficientes de dois modelos de regressão linear. Ela transforma o dia da semana em variáveis dummies, usa o valor e o horário do pedido para estimar o tempo de entrega e, em seguida, utiliza esse tempo previsto como entrada para calcular o NPS. Ao final, retorna o tempo de entrega estimado (em minutos) e o NPS previsto para o pedido.

```
------------------------------------------------------
          ✨ Previsão de Performance do Pedido ✨
------------------------------------------------------
Qual o valor total do pedido (R$)? 30

Dias da semana: (0=Dom, 1=Seg, ..., 6=Sáb)
Qual o número do dia da semana (0 a 6)? 20
Número inválido. Insira um número de 0 a 6.
Qual o número do dia da semana (0 a 6)? 1

--- Horário do Pedido ---
Que horas o pedido foi feito (0 a 23)? 20
E quantos minutos (0 a 59)? 30

------------------------------------------------------
Pedido de R$30.00 | Segunda-feira às 20:30
------------------------------------------------------
⏳ Tempo de Entrega Previsto: 21.95 minutos
☀️ NPS Score Previsto: 61.83
------------------------------------------------------
```

Após a criação de uma função de prompt obtemos um simulador de NPS e Tempo de Entrega que utiliza os dados de entrada de Dia da Semana, Hora, Minuto e Valor do pedido para atribuir seus resultados.

Conclusão

O desenvolvimento dos modelos de regressão linear permitiu compreender com maior clareza os fatores que influenciam o tempo de entrega e a satisfação do cliente (NPS). Através da análise dos dados e da aplicação do método dos mínimos quadrados via decomposição matricial, foi possível construir modelos matemáticos capazes de gerar previsões consistentes e interpretáveis.

Os resultados indicaram uma correlação negativa entre o tempo de entrega e o NPS, mostrando que entregas mais rápidas tendem a elevar a satisfação dos clientes. Além disso, o dia da semana apresentou influência moderada sobre o tempo de entrega, sugerindo variações operacionais conforme o período. O valor do pedido mostrou correlação baixa, indicando que seu impacto sobre o tempo de entrega e o NPS é limitado.