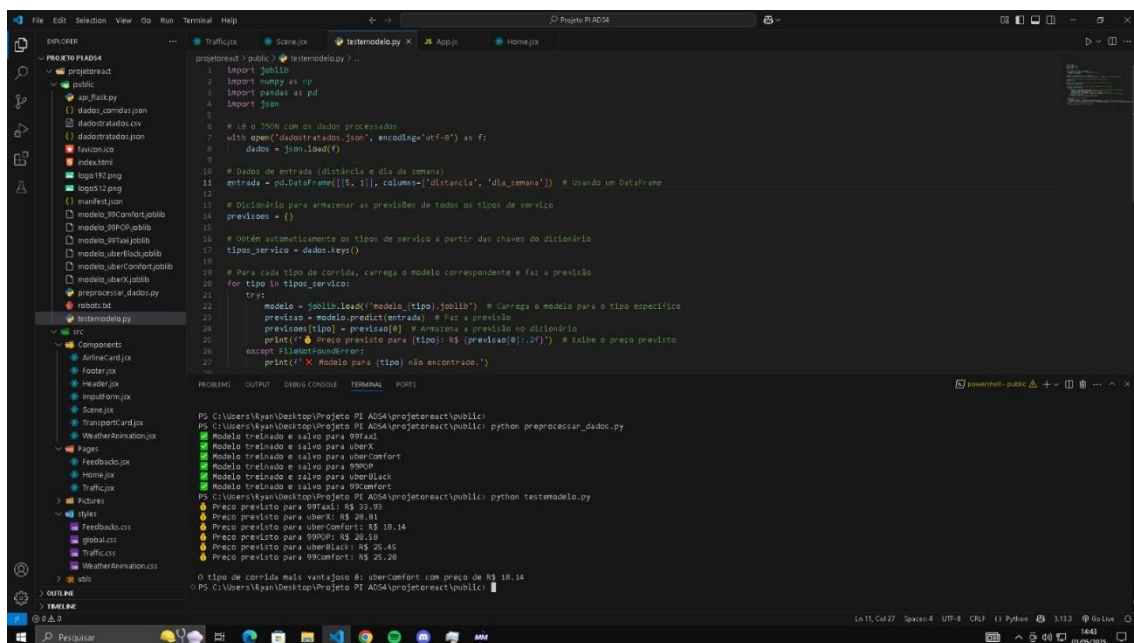


IA e Machine Learning

Neste projeto, implementamos um modelo de Machine Learning baseado no algoritmo **Gradient Boosting Regressor** para prever os preços de serviços com base na distância entre dois pontos. Essa abordagem se mostrou eficiente para resolver problemas de regressão com boa acurácia e robustez.

O que é o Gradient Boosting Regressor?

O **Gradient Boosting Regressor** é um algoritmo de aprendizado de máquina do tipo ensemble que combina múltiplas árvores de decisão fracas (shallow trees) em uma sequência, onde cada nova árvore tenta corrigir os erros da anterior. Esse método é conhecido por sua alta performance em tarefas de previsão contínua (regressão), pois consegue capturar relações complexas entre as variáveis.



```
projectreact > public > testmodelo.py ...
1 import joblib
2 import numpy as np
3 import pandas as pd
4 import json
5
6 # Lê o JSON com os dados processados
7 with open('dados tratados.json', encoding='utf-8') as f:
8     dados = json.load(f)
9
10 # Dados de entrada (distância e dia da semana)
11 entrada = pd.DataFrame([5, 1], columns=['distancia', 'dia_semana']) # Usando um DataFrame
12
13 # Dicionário para armazenar as previsões de todos os tipos de serviço
14 previsoes = {}
15
16 # Ordena automaticamente os tipos de serviço a partir das chaves do dicionário
17 tipos_servico = dados.keys()
18
19 # Para cada tipo de corrida, carrega o modelo correspondente e faz a previsão
20 for tipo in tipos_servico:
21     try:
22         modelo = joblib.load(f'modelo_{tipo}.joblib') # Carrega o modelo para o tipo específico
23         previsao = modelo.predict(entrada) # Faz a previsão
24         previsoes[tipo] = previsao[0] # Armazena a previsão no dicionário
25         print(f'Preço previsto para (tipo): R$ {previsao[0]:.2f}') # Exibe o preço previsto
26     except FileNotFoundError:
27         print(f'X Modelo para (tipo) não encontrado.')
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

```
PS C:\Users\kyan\Desktop\Projeto IA ADSA\projectreact\public> python preprocessor_dados.py
Modelo treinado e salvo para 99taxi
Modelo treinado e salvo para uberX
Modelo treinado e salvo para uberComfort
Modelo treinado e salvo para 99POP
Modelo treinado e salvo para uberBlack
Modelo treinado e salvo para 99Comfort
PS C:\Users\kyan\Desktop\Projeto IA ADSA\projectreact\public> python testmodelo.py
Preço previsto para 99taxi: R$ 33.90
Preço previsto para uberX: R$ 29.81
Preço previsto para uberComfort: R$ 19.14
Preço previsto para 99POP: R$ 23.15
Preço previsto para uberBlack: R$ 25.45
Preço previsto para 99Comfort: R$ 25.28
O tipo de corrida mais vantajoso é: uberComfort com preço de R$ 19.14
PS C:\Users\kyan\Desktop\Projeto IA ADSA\projectreact\public>
```

Após o treinamento, o modelo foi integrado ao back-end Flask. O front-end do site consome a API para:

1. Enviar os dados de entrada (ponto A até ponto B);
2. O back-end processa esses dados com o modelo;
3. A API retorna os valores previstos para cada serviço e destaca a opção mais econômica.

