```
import sys
!{sys.executable} -m pip install geopy
```

```
Requirement already satisfied: geopy in
c:\users\mathe\anaconda3\lib\site-packages (2.4.1)
Requirement already satisfied: geographiclib<3,>=1.52 in
c:\users\mathe\anaconda3\lib\site-packages (from geopy) (2.0)
```

```
import pandas as pd
from geopy.distance import geodesic
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error,
        r2_score
import numpy as np


# 1. Carregamento dos arquivos
dfRide = pd.read_csv("ride_v2.csv", sep=";", dtype=str)
dfRideAdd = pd.read_csv("rideaddress_v1.csv", sep=";", dtype=str)
dfRideEst = pd.read_csv("rideestimative_v3.csv", sep=";", dtype=str)
dfProduct = pd.read_csv("product.csv", sep=";", dtype=str)


# 2. Uniformização: datas e RideID
dfRide["Schedule"] = pd.to_datetime(dfRide["Schedule"],
        errors="coerce")
for df in [dfRide, dfRideAdd, dfRideEst]:
    df["RideID"] = df["RideID"].astype(str).str.replace(".0", "",
        regex=False)


# 3. Derivar colunas de tempo
dfRide["Dia"] = dfRide["Schedule"].dt.weekday
dfRide["Hora"] = dfRide["Schedule"].dt.hour
dfRide["Minuto"] = dfRide["Schedule"].dt.minute
dfRide["HoraDecimal"] = dfRide["Hora"] + dfRide["Minuto"] / 60
dfRide["Faixa15min"] = dfRide["Schedule"].dt.floor("15min")
dfTempo = dfRide[["RideID", "Dia", "Hora", "Minuto", "HoraDecimal",
        "Faixa15min"]].dropna()


# 4. Extrair origem e destino (Lat, Lng, Address)
dfRideAdd = dfRideAdd.rename(columns={"RideAddressTypeID":
        "OrigDest"})
dfOrigem = dfRideAdd[dfRideAdd["OrigDest"] == "1"][["RideID", "Lat",
        "Lng", "Address"]].rename(
    columns={"Lat": "Lat1", "Lng": "Lng1", "Address": "AddressOrig"}
)
dfDestino = dfRideAdd[dfRideAdd["OrigDest"] == "2"][["RideID", "Lat",
        "Lng", "Address"]].rename(
    columns={"Lat": "Lat2", "Lng": "Lng2", "Address": "AddressDest"}
)
dfCoords = pd.merge(dfOrigem, dfDestino, on="RideID", how="inner")


# Corrige vírgulas e converte coordenadas
for col in ["Lat1", "Lng1", "Lat2", "Lng2"]:
    dfCoords[col] = dfCoords[col].str.replace(",",
        ".").astype(float).round(6)


# 5. Integrar todas as estimativas com produtos em UMA COLUNA tipo
        dicionário
dfRideEst["ProductID"] = dfRideEst["ProductID"].astype(str)
dfProduct["ProductID"] = dfProduct["ProductID"].astype(str)


dfEstimadaComProduto = pd.merge(dfRideEst, dfProduct, on="ProductID",
        how="left")
dfEstimadaComProduto["Price"] =
        dfEstimadaComProduto["Price"].str.replace(",",
        ".").astype(float)


dfEstimadaSelecionada = dfEstimadaComProduto.groupby("RideID").apply(
    lambda x: dict(zip(x["Description"], x["Price"]))
```

```
).reset_index().rename(columns={0: "Estimativas"})

# 6. Refiltra pelos RideID em comum
dfCoords["RideID"] = dfCoords["RideID"].astype(str)
dfEstimadaSelecionada["RideID"] =
        dfEstimadaSelecionada["RideID"].astype(str)
ids_comuns = set(dfTempo["RideID"]) & set(dfCoords["RideID"]) &
        set(dfEstimadaSelecionada["RideID"])

dfTempo =
        dfTempo[dfTempo["RideID"].isin(ids_comuns)].sort_values("RideID").reset_index(drop=True)
dfCoords =
        dfCoords[dfCoords["RideID"].isin(ids_comuns)].sort_values("RideID").reset_index(drop=True)
dfEstimadaSelecionada =
        dfEstimadaSelecionada[dfEstimadaSelecionada["RideID"].isin(ids_comuns)].sort_values("RideID"

# 7. Junta tudo sem merge (concatenando os DataFrames horizontalmente)
dfDerivado = pd.concat([
    dfTempo,
    dfCoords.drop(columns=["RideID"]),
    dfEstimadaSelecionada.drop(columns=["RideID"])
], axis=1)

# 8. Remove NaNs nas coordenadas
dfDerivado = dfDerivado.dropna(subset=["Lat1", "Lng1", "Lat2",
        "Lng2"]).reset_index(drop=True)

# 9. Cálculo da distância
dfDerivado["Distancia_km"] = dfDerivado.apply(
    lambda row: geodesic((row["Lat1"], row["Lng1"]), (row["Lat2"],
        row["Lng2"])).kilometers,
    axis=1
)

# Mostrar amostra do DataFrame final com estimativas e valor_uber
print("\n📊 Exemplo de dados do dfDerivado com estimativas:")
print(dfDerivado.head(10))
```

```
C:\Users\mathe\AppData\Local\Temp\ipykernel_18272\3639340257.py:48:
DeprecationWarning: DataFrameGroupBy.apply operated on the grouping
columns. This behavior is deprecated, and in a future version of
pandas the grouping columns will be excluded from the operation.
Either pass `include_groups=False` to exclude the groupings or
explicitly select the grouping columns after groupby to silence this
warning.
  dfEstimadaSelecionada =
dfEstimadaComProduto.groupby("RideID").apply(


📊 Exemplo de dados do dfDerivado com estimativas:
    RideID  Dia  Hora  Minuto  HoraDecimal          Faixa15min
Lat1  \
0  1183200    1    10       9    10.150000 2021-08-17 10:00:00
-26.329754
1  1183201    1    10       9    10.150000 2021-08-17 10:00:00
-27.491979
2  1183202    1    10      10    10.166667 2021-08-17 10:00:00
-19.849580
3  1183203    1    10      10    10.166667 2021-08-17 10:00:00
-23.962423
4  1183204    1    10      10    10.166667 2021-08-17 10:00:00
-10.919802
5  1183205    1    10      10    10.166667 2021-08-17 10:00:00
-22.873502
6  1183206    1    10      10    10.166667 2021-08-17 10:00:00
-23.554281
7  1183207    1    10      10    10.166667 2021-08-17 10:00:00
-23.962423
8  1183208    1    10      10    10.166667 2021-08-17 10:00:00
```

```
-19.849539
9  1183209    1    10      10    10.166667 2021-08-17 10:00:00
-8.025771

        Lng1                                        AddressOrig
Lat2  \
0 -48.840428  Rua João Pinheiro, 585 - Rua João Pinheiro - B...
-26.255466
1 -48.528288  Rodovia Rafael da Rocha Pires, 1883 - Rodovia ...
-27.437149
2 -44.019916  Rua Barão do Rio Branco, 12 - Rua Barão do Rio...
-19.936899
3 -46.254658               Tv. Duzentos e Sessenta e Um, 72, 72
-23.837307
4 -37.077442        Rua Argentina, 160 - Rua Argentina - Brasil
-10.907129
5 -43.571402  Rua João Cirílo de Oliveira, 5 - Rua João Cirí...
-22.917373
6 -46.662732  Avenida Angélica, 2573 - Avenida Angélica - Br...
-23.734290
7 -46.254658               Tv. Duzentos e Sessenta e Um, 72, 72
-23.837307
8 -44.019929  R. Barão do Rio Branco, 12 - Nacional, Contage...
-19.936899
9 -34.874475   Rua João Alfredo, 83 - Rua João Alfredo - Brasil
-8.029684

        Lng2                                        AddressDest  \
0 -48.643420  Av. Dr. Nereu Ramos, 450 - Rocio Grande, São F...
1 -48.398243  Angeloni Ingleses (Florianópolis) - Supermerca...
2 -43.940160  R. Antônio de Albuquerque, 1080 - Funcionários...
3 -46.132172                 Semar Supermercados Bertioga, 2141
4 -37.087719  R. Simeão Aguiar, 430 - Novo Paraíso, Aracaju ...
5 -43.633044  Av. Cesário de Melo, 10809 - Paciência, Rio de...
6 -46.698628  Av. Sen. Teotônio Vilela, 4029 - Parque Cocaia...
7 -46.132172                 Semar Supermercados Bertioga, 2141
8 -43.940160  R. Antônio de Albuquerque, 1080 - Funcionários...
9 -34.944037  Av. Professor Joaquim Cavalcanti, 721 - Iputin...

                                         Estimativas  Distancia_km
0  {'Flash': 89.0, 'UberX': 89.0, 'Comfort': 116....     21.327034
1  {'Flash': 31.5, 'Comfort': 33.5, '99POUPA': 26...     14.217724
2  {'Moto': 25.5, 'Comfort': 44.0, 'UberFlash': 4...     12.774740
3  {'Flash': 47.5, '99POP': 63.69, 'Comfort': 68....     18.644013
4  {'99POUPA': 7.88, '99POP': 7.88, '99TAXI': 17....      1.796461
5  {'99POUPA': 14.69, '99POP': 19.51, '99TAXI': 5...      7.975203
6  {'UberX': 46.5, 'Comfort': 71.5, 'Flash': 43.0...     20.270171
7  {'99TAXI': 118.16, 'Táxi Comum': 138.95, 'Flas...     18.644013
8  {'99POUPA': 71.3, '99POP': 71.3, '99TAXI': 62....     12.779065
9  {'Flash': 18.0, 'Flash Moto': 12.0, 'Comfort':...      7.680426
```

```python
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error,
        r2_score
import numpy as np

# 🚚 Lista de serviços da Uber a prever
servicos_alvo = ["UberX", "Comfort", "Black"]

print("📈 Treinando modelos para serviços da Uber (com features
        auxiliares):\n")

# Lista para armazenar os resultados
resultados = []
```

```
for servico in servicos_alvo:
    print(f"🔮 Iniciando modelo para: {servico}")

    # Filtra somente onde o serviço alvo tem valor
    df_modelo = dfDerivado[dfDerivado["Estimativas"].apply(
        lambda d: isinstance(d, dict) and servico in d and
        isinstance(d[servico], (int, float))
    )].copy()

    if df_modelo.empty:
        print(f"⚠️ Nenhum dado encontrado para {servico}.
        Ignorado.\n")
        continue

    # Define a variável alvo
    df_modelo["y"] = df_modelo["Estimativas"].apply(lambda d:
        d[servico])

    # Features base
    features_base = ["Distancia_km", "Dia", "Hora", "HoraDecimal",
        "Lat1", "Lng1", "Lat2", "Lng2"]

    # Adiciona colunas auxiliares com os demais serviços (exceto o
        alvo)
    servicos_aux = set()
    df_modelo["Estimativas"].apply(lambda d:
        servicos_aux.update(d.keys()) if isinstance(d, dict) else
        None)
    servicos_aux.discard(servico)

    for s in servicos_aux:
        nome_coluna = f"aux_{s.lower().replace(' ', '_')}"
        df_modelo[nome_coluna] = df_modelo["Estimativas"].apply(lambda
        d: d.get(s) if isinstance(d, dict) else np.nan)

    # Prepara X e y
    features_auxiliares = [col for col in df_modelo.columns if
        col.startswith("aux_")]
    X = df_modelo[features_base + features_auxiliares].fillna(-1)
    y = df_modelo["y"]

    # Verificação mínima de dados
    if len(X) < 100:
        print(f"⚠️ Serviço '{servico}' com poucos dados após
        preparação ({len(X)} linhas) — ignorado.\n")
        continue

    # Treinamento
    X_train, X_test, y_train, y_test = train_test_split(X, y,
        test_size=0.2, random_state=42)
    modelo = RandomForestRegressor(random_state=42)
    modelo.fit(X_train, y_train)
    y_pred = modelo.predict(X_test)

    # Métricas
    mae = mean_absolute_error(y_test, y_pred)
    rmse = np.sqrt(mean_squared_error(y_test, y_pred))
    r2 = r2_score(y_test, y_pred)

    resultados.append({
        "Serviço": servico,
        "Registros": len(X),
        "MAE": round(mae, 2),
        "RMSE": round(rmse, 2),
        "R²": round(r2, 4)
    })

    print(f"✅ Modelo treinado para: {servico}")
    print(f" → MAE  : R${mae:.2f}")
    print(f" → RMSE : R${rmse:.2f}")
    print(f" → R²   : {r2:.4f}\n")

# 🔚 Resumo final
if resultados:
    df_resultados = pd.DataFrame(resultados).sort_values(by="R²",
        ascending=False)
    print("📊 Resumo Final dos Modelos:")
    print(df_resultados.to_string(index=False))
else:
    print("❌ Nenhum modelo foi treinado.")
```

```
📈 Treinando modelos para serviços da Uber (com features auxiliares):

🔮 Iniciando modelo para: UberX
✅ Modelo treinado para: UberX
 → MAE  : R$0.48
 → RMSE : R$2.26
 → R²   : 0.9905

🔮 Iniciando modelo para: Comfort
✅ Modelo treinado para: Comfort
 → MAE  : R$2.20
 → RMSE : R$4.87
 → R²   : 0.9808

🔮 Iniciando modelo para: Black
✅ Modelo treinado para: Black
 → MAE  : R$1.82
 → RMSE : R$4.86
 → R²   : 0.9852

📊 Resumo Final dos Modelos:
Serviço  Registros  MAE  RMSE     R²
  UberX     235601 0.48  2.26 0.9905
  Black     123666 1.82  4.86 0.9852
Comfort     192876 2.20  4.87 0.9808
```

```
import sys
!{sys.executable} -m pip install dash dash-bootstrap-components
        jupyter-dash
```

```
Requirement already satisfied: dash in
c:\users\mathe\anaconda3\lib\site-packages (3.0.4)
Requirement already satisfied: dash-bootstrap-components in
c:\users\mathe\anaconda3\lib\site-packages (2.0.2)
Requirement already satisfied: jupyter-dash in
c:\users\mathe\anaconda3\lib\site-packages (0.4.2)
Requirement already satisfied: Flask<3.1,>=1.0.4 in
c:\users\mathe\anaconda3\lib\site-packages (from dash) (3.0.3)
Requirement already satisfied: Werkzeug<3.1 in
c:\users\mathe\anaconda3\lib\site-packages (from dash) (3.0.3)
Requirement already satisfied: plotly>=5.0.0 in
c:\users\mathe\anaconda3\lib\site-packages (from dash) (5.24.1)
Requirement already satisfied: importlib-metadata in
c:\users\mathe\anaconda3\lib\site-packages (from dash) (7.0.1)
Requirement already satisfied: typing-extensions>=4.1.1 in
c:\users\mathe\anaconda3\lib\site-packages (from dash) (4.11.0)
Requirement already satisfied: requests in
c:\users\mathe\anaconda3\lib\site-packages (from dash) (2.32.3)
Requirement already satisfied: retrying in
c:\users\mathe\anaconda3\lib\site-packages (from dash) (1.3.4)
Requirement already satisfied: nest-asyncio in
c:\users\mathe\anaconda3\lib\site-packages (from dash) (1.6.0)
Requirement already satisfied: setuptools in
c:\users\mathe\anaconda3\lib\site-packages (from dash) (75.1.0)
Requirement already satisfied: ipython in
c:\users\mathe\anaconda3\lib\site-packages (from jupyter-dash)
```

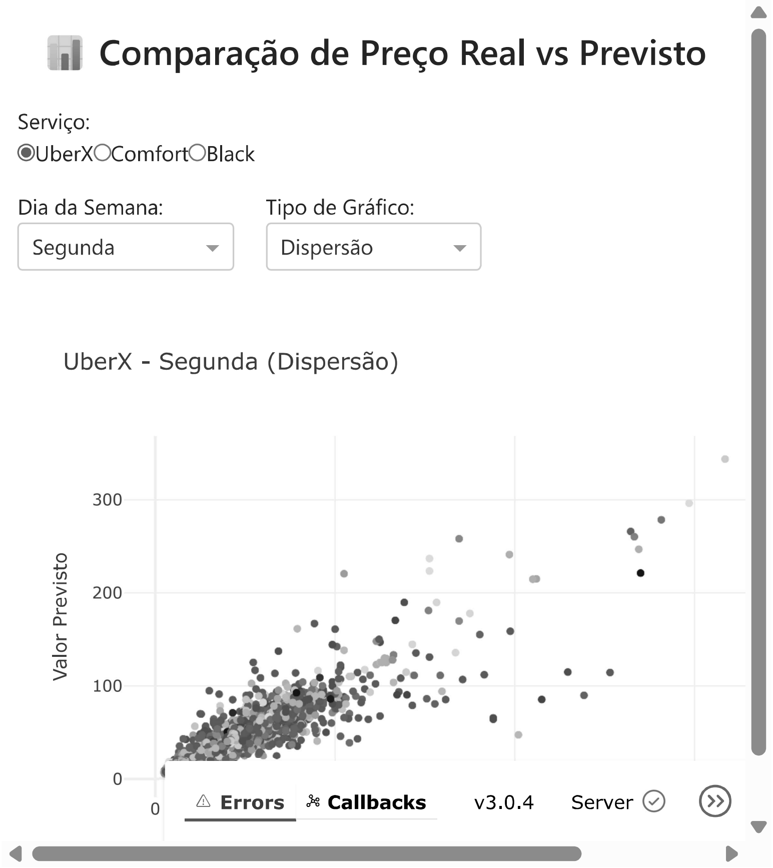
```
(8.27.0)
Requirement already satisfied: ipykernel in
c:\users\mathe\anaconda3\lib\site-packages (from jupyter-dash)
(6.28.0)
Requirement already satisfied: ansi2html in
c:\users\mathe\anaconda3\lib\site-packages (from jupyter-dash) (1.9.2)
Requirement already satisfied: Jinja2>=3.1.2 in
c:\users\mathe\anaconda3\lib\site-packages (from Flask<3.1,>=1.0.4-
>dash) (3.1.4)
Requirement already satisfied: itsdangerous>=2.1.2 in
c:\users\mathe\anaconda3\lib\site-packages (from Flask<3.1,>=1.0.4-
>dash) (2.2.0)
Requirement already satisfied: click>=8.1.3 in
c:\users\mathe\anaconda3\lib\site-packages (from Flask<3.1,>=1.0.4-
>dash) (8.1.7)
Requirement already satisfied: blinker>=1.6.2 in
c:\users\mathe\anaconda3\lib\site-packages (from Flask<3.1,>=1.0.4-
>dash) (1.6.2)
Requirement already satisfied: tenacity>=6.2.0 in
c:\users\mathe\anaconda3\lib\site-packages (from plotly>=5.0.0->dash)
(8.2.3)
Requirement already satisfied: packaging in
c:\users\mathe\anaconda3\lib\site-packages (from plotly>=5.0.0->dash)
(24.1)
Requirement already satisfied: MarkupSafe>=2.1.1 in
c:\users\mathe\anaconda3\lib\site-packages (from Werkzeug<3.1->dash)
(2.1.3)
Requirement already satisfied: zipp>=0.5 in
c:\users\mathe\anaconda3\lib\site-packages (from importlib-metadata-
>dash) (3.17.0)
Requirement already satisfied: comm>=0.1.1 in
c:\users\mathe\anaconda3\lib\site-packages (from ipykernel->jupyter-
dash) (0.2.1)
Requirement already satisfied: debugpy>=1.6.5 in
c:\users\mathe\anaconda3\lib\site-packages (from ipykernel->jupyter-
dash) (1.6.7)
Requirement already satisfied: jupyter-client>=6.1.12 in
c:\users\mathe\anaconda3\lib\site-packages (from ipykernel->jupyter-
dash) (8.6.0)
Requirement already satisfied: jupyter-core!=5.0.*,>=4.12 in
c:\users\mathe\anaconda3\lib\site-packages (from ipykernel->jupyter-
dash) (5.7.2)
Requirement already satisfied: matplotlib-inline>=0.1 in
c:\users\mathe\anaconda3\lib\site-packages (from ipykernel->jupyter-
dash) (0.1.6)
Requirement already satisfied: psutil in
c:\users\mathe\anaconda3\lib\site-packages (from ipykernel->jupyter-
dash) (5.9.0)
Requirement already satisfied: pyzmq>=24 in
c:\users\mathe\anaconda3\lib\site-packages (from ipykernel->jupyter-
dash) (25.1.2)
Requirement already satisfied: tornado>=6.1 in
c:\users\mathe\anaconda3\lib\site-packages (from ipykernel->jupyter-
dash) (6.4.1)
Requirement already satisfied: traitlets>=5.4.0 in
c:\users\mathe\anaconda3\lib\site-packages (from ipykernel->jupyter-
dash) (5.14.3)
Requirement already satisfied: decorator in
c:\users\mathe\anaconda3\lib\site-packages (from ipython->jupyter-
dash) (5.1.1)
Requirement already satisfied: jedi>=0.16 in
c:\users\mathe\anaconda3\lib\site-packages (from ipython->jupyter-
dash) (0.19.1)
Requirement already satisfied: prompt-toolkit<3.1.0,>=3.0.41 in
c:\users\mathe\anaconda3\lib\site-packages (from ipython->jupyter-
dash) (3.0.43)
Requirement already satisfied: pygments>=2.4.0 in
c:\users\mathe\anaconda3\lib\site-packages (from ipython->jupyter-
dash) (2.15.1)
Requirement already satisfied: stack-data in
c:\users\mathe\anaconda3\lib\site-packages (from ipython->jupyter-
dash) (0.2.0)
Requirement already satisfied: colorama in
c:\users\mathe\anaconda3\lib\site-packages (from ipython->jupyter-
dash) (0.4.6)
Requirement already satisfied: charset-normalizer<4,>=2 in
c:\users\mathe\anaconda3\lib\site-packages (from requests->dash)
(3.3.2)
Requirement already satisfied: idna<4,>=2.5 in
c:\users\mathe\anaconda3\lib\site-packages (from requests->dash) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in
c:\users\mathe\anaconda3\lib\site-packages (from requests->dash)
(2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in
c:\users\mathe\anaconda3\lib\site-packages (from requests->dash)
(2024.8.30)
Requirement already satisfied: six>=1.7.0 in
c:\users\mathe\anaconda3\lib\site-packages (from retrying->dash)
(1.16.0)
Requirement already satisfied: parso<0.9.0,>=0.8.3 in
c:\users\mathe\anaconda3\lib\site-packages (from jedi>=0.16->ipython-
>jupyter-dash) (0.8.3)
Requirement already satisfied: python-dateutil>=2.8.2 in
c:\users\mathe\anaconda3\lib\site-packages (from jupyter-
client>=6.1.12->ipykernel->jupyter-dash) (2.9.0.post0)
Requirement already satisfied: platformdirs>=2.5 in
c:\users\mathe\anaconda3\lib\site-packages (from jupyter-
core!=5.0.*,>=4.12->ipykernel->jupyter-dash) (3.10.0)
Requirement already satisfied: pywin32>=300 in
c:\users\mathe\anaconda3\lib\site-packages (from jupyter-
core!=5.0.*,>=4.12->ipykernel->jupyter-dash) (305.1)
Requirement already satisfied: wcwidth in
c:\users\mathe\anaconda3\lib\site-packages (from prompt-
toolkit<3.1.0,>=3.0.41->ipython->jupyter-dash) (0.2.5)
Requirement already satisfied: executing in
c:\users\mathe\anaconda3\lib\site-packages (from stack-data->ipython-
>jupyter-dash) (0.8.3)
Requirement already satisfied: asttokens in
c:\users\mathe\anaconda3\lib\site-packages (from stack-data->ipython-
>jupyter-dash) (2.0.5)
Requirement already satisfied: pure-eval in
c:\users\mathe\anaconda3\lib\site-packages (from stack-data->ipython-
>jupyter-dash) (0.2.2)
```

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
import numpy as np
import pandas as pd


# Simular dfDerivado com colunas mínimas
nomes_dias = ['Segunda', 'Terça', 'Quarta', 'Quinta', 'Sexta',
        'Sábado', 'Domingo']
servicos = ["UberX", "Comfort", "Black"]

# Aqui está um exemplo base que você substituirá pelo seu dfDerivado
        real:
dfDerivado = dfDerivado.copy()  # <- use seu dfDerivado real aqui


df_dashboard_final = pd.DataFrame()


for servico in servicos:
    df_modelo = dfDerivado[dfDerivado["Estimativas"].apply(
        lambda d: isinstance(d, dict) and servico in d and
        isinstance(d[servico], (int, float))
    )].copy()

    if df_modelo.empty:
        continue

    df_modelo["y"] = df_modelo["Estimativas"].apply(lambda d:
        d[servico])
    y = df_modelo["y"]

    features_base = ["Distancia_km", "Dia", "Hora", "HoraDecimal",
        "Lat1", "Lng1", "Lat2", "Lng2"]
    X = df_modelo[features_base].fillna(-1)

    X_train, X_test, y_train, y_test = train_test_split(X, y,
        test_size=0.2, random_state=42)

    modelo = RandomForestRegressor(random_state=42)
    modelo.fit(X_train, y_train)
    y_pred = modelo.predict(X_test)

    df_resultado = X_test.copy()
    df_resultado["valor_real"] = y_test
    df_resultado["valor_previsto"] = y_pred
    df_resultado["servico"] = servico
    df_resultado["hora"] = df_resultado["Hora"]
    df_resultado["dia_nome"] =
        df_resultado["Dia"].map(dict(enumerate(nomes_dias)))

    df_dashboard_final = pd.concat([df_dashboard_final, df_resultado],
        ignore_index=True)
    df_dashboard_final.head(10)
```

```
from jupyter_dash import JupyterDash
from dash import dcc, html, Input, Output
import dash_bootstrap_components as dbc
import plotly.express as px

# df_dashboard_final deve estar carregado do código anterior

app = JupyterDash(__name__, external_stylesheets=
        [dbc.themes.BOOTSTRAP])

servicos_disponiveis = df_dashboard_final["servico"].unique()
dias_disponiveis = df_dashboard_final["dia_nome"].unique()

app.layout = dbc.Container([
    html.H2("📊 Comparação de Preço Real vs Previsto",
        className="text-center my-4"),

    dbc.Row([
        dbc.Col([
            html.Label("Serviço:"),
            dcc.RadioItems(
                id='filtro_servico',
                options=[{'label': s, 'value': s} for s in
        servicos_disponiveis],
                value=servicos_disponiveis[0],
                inline=True
            )
        ], width=12)
    ], className="mb-3"),

    dbc.Row([
        dbc.Col([
            html.Label("Dia da Semana:"),
            dcc.Dropdown(
                id='filtro_dia',
                options=[{'label': d, 'value': d} for d in
        dias_disponiveis],
                value='Segunda',
                clearable=False
            )
        ], width=4),

        dbc.Col([
            html.Label("Tipo de Gráfico:"),
            dcc.Dropdown(
                id='tipo_grafico',
                options=[
                    {'label': 'Dispersão', 'value': 'dispersao'},
                    {'label': 'Barras', 'value': 'barras'}
                ],
                value='dispersao',
                clearable=False
            )
        ], width=4)
    ], className="mb-4"),

    dcc.Graph(id='grafico_resultado')

], fluid=True)

@app.callback(
    Output('grafico_resultado', 'figure'),
    [Input('filtro_servico', 'value'),
     Input('filtro_dia', 'value'),
     Input('tipo_grafico', 'value')]
)
def atualizar_grafico(servico, dia, tipo_grafico):
    df_filtrado = df_dashboard_final[
        (df_dashboard_final['servico'] == servico) &
        (df_dashboard_final['dia_nome'] == dia)
    ]

    if tipo_grafico == 'dispersao':
        fig = px.scatter(
            df_filtrado, x='valor_real', y='valor_previsto',
            color='hora',
            labels={'valor_real': 'Valor Real', 'valor_previsto':
        'Valor Previsto'},
            title=f"{servico} - {dia} (Dispersão)",
            hover_data=['hora']
        )
    else:
        df_bar = df_filtrado.reset_index(drop=True).head(20)
        fig = px.bar(
            df_bar, x=df_bar.index,
            y=['valor_real', 'valor_previsto'],
            barmode='group',
            labels={'value': 'Valor', 'index': 'Corrida'},
            title=f"{servico} - {dia} (Barras)"
        )

    fig.update_layout(template='plotly_white')
    return fig

# Rodar o app
app.run(mode='inline', debug=True)
```

```
C:\Users\mathe\anaconda3\Lib\site-packages\dash\dash.py:587:
UserWarning:


JupyterDash is deprecated, use Dash instead.
```

```
See https://dash.plotly.com/dash-in-jupyter for more details.
```



📊 Comparação de Preço Real vs Previsto

Serviço:
● UberX ○ Comfort ○ Black

Dia da Semana: Segunda

Tipo de Gráfico: Dispersão

```
pip install pandoc
```

```
Requirement already satisfied: pandoc in
c:\users\mathe\anaconda3\lib\site-packages (2.4)
Requirement already satisfied: plumbum in
c:\users\mathe\anaconda3\lib\site-packages (from pandoc) (1.9.0)
Requirement already satisfied: ply in
c:\users\mathe\anaconda3\lib\site-packages (from pandoc) (3.11)
Requirement already satisfied: pywin32 in
c:\users\mathe\anaconda3\lib\site-packages (from plumbum->pandoc)
(305.1)
Note: you may need to restart the kernel to use updated packages.
```