Vinícius Miranda Andrade Piovesan RA: 23025544

Felipe Ribeiro Almeida RA: 23024683

Sérgio Ricardo Pedote Junior RA:23747441

MATHEUS DE MEDEIROS TAKAKI RA: 23025143

Documentação Técnica – Modelos de Machine Learning e IA no Projeto

1. Modelo de Machine Learning Supervisionado — Random Forest Regressor

Objetivo:

Prever o preço estimado de um serviço com base em dados históricos de distância e duração da rota.

Como funciona:

- **Entrada:** duas variáveis numéricas — distância (km) e duração (minutos).
- **Saída:** preço estimado.
- O modelo foi treinado usando a técnica **Random Forest Regressor**, que é um conjunto de múltiplas árvores de decisão (ensemble learning).
- Cada árvore faz uma previsão, e a média das previsões é o resultado final.
- É um modelo robusto, que lida bem com dados não lineares e ruído.

Implementação:

- Os dados são carregados de arquivos CSV por categoria.
- O modelo é treinado para cada categoria e salvo em disco para uso posterior.
- No backend Flask, esses modelos são carregados e usados para prever o preço dado origem e destino do usuário.

Vantagens:

- Alta precisão em dados tabulares.
- Fácil de interpretar e ajustar.

- Bom controle sobre overfitting via parâmetros.

---

2. Modelo de Inteligência Artificial Não Supervisionado — KMeans Clustering

Objetivo:

Agrupar usuários com base em seu comportamento no sistema para segmentação e análise.

Como funciona:

- Recebe dados quantitativos de comportamento, por exemplo: número de acessos, duração média da sessão, páginas visitadas.
- Aplica o algoritmo **KMeans**, que agrupa os usuários em clusters com características semelhantes.
- O algoritmo define a posição central de cada grupo (centroide) e atribui cada usuário ao cluster mais próximo.

Implementação:

- Os dados de comportamento são armazenados no arquivo comportamento.csv.
- O script treinar_cluster.py carrega esses dados, normaliza e executa o KMeans para formar os grupos.
- O número de clusters (n_clusters) pode ser ajustado conforme necessidade.

Vantagens:

- Descobre padrões sem necessidade de rótulos (não supervisionado).
- Útil para segmentar usuários para ações personalizadas.
- Rápido e eficiente para conjuntos de dados pequenos a médios.

Conclusão

O projeto integra **modelos supervisionados e não supervisionados** para oferecer funcionalidades avançadas:

- O **Random Forest** fornece previsões de preço confiáveis para rotas.
- O **KMeans** possibilita análises de comportamento e segmentação de usuários.

Essa combinação agrega valor ao sistema, facilitando tanto a automação das previsões quanto a personalização via IA.

ML:

```
import pandas as pd
import os
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import r2_score, mean_squared_error
import joblib
import numpy as np

caminho_categorias = "categorias"
arquivos = os.listdir(caminho_categorias)


os.makedirs("modelos", exist_ok=True)

for arquivo in arquivos:
  categoria = arquivo.replace(".csv", "")
  caminho_csv = os.path.join(caminho_categorias, arquivo)
  df = pd.read_csv(caminho_csv, sep=";")


  df = df.dropna(subset=["Distancia_km", "Duracao_min", "Price"])

  X = df[["Distancia_km", "Duracao_min"]]
  y = df["Price"]


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


  param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5],
```

```
    'min_samples_leaf': [1, 2]
}

modelo_base = RandomForestRegressor(random_state=42, n_jobs=-1)
grid_search = GridSearchCV(
    estimator=modelo_base,
    param_grid=param_grid,
    cv=3,
    scoring='r2',
    n_jobs=-1,
    verbose=1
)

grid_search.fit(X_train, y_train)
melhor_modelo = grid_search.best_estimator_

y_pred = melhor_modelo.predict(X_test)
r2 = r2_score(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))

print(f" Avaliação do modelo '{categoria}':")
print(f"    Melhor params: {grid_search.best_params_}")
print(f"    R²     : {r2:.4f}")
print(f"    RMSE   : {rmse:.2f}")

nome_modelo = f"modelos/model_{categoria}.pkl"
joblib.dump(melhor_modelo, nome_modelo)
print(f"✅ Modelo salvo: {nome_modelo}\n")
```

IA:

```
usuarios = carregar_usuarios()
for u in usuarios:
  if u["usuario"] == usuario and bcrypt.check_password_hash(u["senhaHash"],
senha):
    # === REGISTRA COMPORTAMENTO SIMULADO ===
    import numpy as np

    acessos = np.random.randint(1, 30)
    duracao_sessao = round(np.random.uniform(2, 60), 2)
    paginas_visitadas = np.random.randint(1, 10)

    linha = f"{usuario},{acessos},{duracao_sessao},{paginas_visitadas}\n"
    with open(COMPORTAMENTO_CSV, "a") as f:
      f.write(linha)

    if os.path.exists(MODELO_CLUSTER_PATH):
      modelo = joblib.load(MODELO_CLUSTER_PATH)
      grupo = int(modelo.predict([[acessos, duracao_sessao,
paginas_visitadas]])[0])
    else:
      grupo = "Modelo não treinado"
```