FUNDAÇÃO ESCOLA DE COMÉRCIO ALVARES PENTEADO

# Algoritmo inicial
# Fasor - Fare Advisor

SÃO PAULO

**2025**

**Integrantes:**

Isaac Santos RA: 23025417

Caroline Gomes RA: 23024619

Icaro Luiz RA: 23025413

Giovanne Braga RA: 23025648

# Sumário

**Introdução** ..... 4
**ALGORITMO PRETENDIDO PARA UTILIZAÇÃO** ..... 5
**BASE DE DADOS** ..... 6
**TREINAMENTO DO MODELO** ..... 7
**RESULTADOS E ANÁLISES** ..... 8
**EXEMPLO DO ALGORITMO DE REGRESSÃO LINEAR:** ..... 9
**CONCLUSÃO** ..... 10

# Introdução

O objetivo do "Fasor - Fare Advisor" é prever os preços de corridas de aplicativos de transporte utilizando Inteligência Artificial (IA). A previsão de preços permite oferecer estimativas mais precisas aos usuários, ajudando na escolha mais interessante para usuário (como preço ou tempo). Para isso, utilizaremos algoritmos de Machine Learning que serão treinados com dados fornecidos pela Khipo de corridas anteriores, considerando variáveis como distância, tempo, meteorologia, horário, demanda e outros fatores relevantes.

# ALGORITMO PRETENDIDO PARA UTILIZAÇÃO

Para a previsão dos preços das corridas, foi escolhido inicialmente o modelo de Regressão Linear. Este modelo foi selecionado devido à sua simplicidade e capacidade de capturar relações lineares entre as variáveis envolvidas. A regressão linear é amplamente utilizada para prever valores contínuos por isso sua escolha como algoritmo principal foi feita.

Porém, ficará em "nossos radares" a implementação do algoritmo "Random Forest", que pode capturar padrões mais complexos nos dados e lidar melhor com outliers e variáveis categóricas. A comparação entre os modelos permitirá determinar qual apresenta o melhor desempenho para a estimativa de preços.

As variáveis mais relevantes serão extraídas da base histórica para a criação do modelo.

# BASE DE DADOS

A base de dados que será utilizada é a base histórica fornecida pela Khipo, ela possui dados de corridas dos principais serviços de aplicativos como Uber e 99, as colunas de origem e destino e serviços utilizados serão os alvos principais para o treinamento do modelo.

Como a base carece de algumas informações como meteorologia e distância, algumas APIs como a do Google Maps, Autocomplete e Accuweather serão utilizadas para complementar as informações faltantes.

# TREINAMENTO DO MODELO

O treinamento do modelo é um desafio, então os dados serão divididos em dois conjuntos seguindo o padrão:

Conjunto de Treinamento (80% dos dados): Utilizado para ajustar os parâmetros do modelo.

Conjunto de Teste (20% dos dados): Utilizado para avaliar a precisão do modelo em previsões não vistas durante o treinamento.

O treinamento será realizado utilizando a biblioteca scikit-learn (1ª opção), onde o modelo de regressão linear será ajustado com os dados de treino. Foi realizada uma análise profunda dos dados fornecidos, esses dados serão ajustados e parametrizados para melhorar a performance do modelo.

Também será colhido feedback por parte dos usuários, para auxiliar no treinamento do modelo.

# RESULTADOS E ANÁLISES

As avalições do modelo se darão por:

Gráfico de previsões vs. valores reais: será gerado um gráfico de dispersão para visualizar a proximidade entre os valores previstos e os valores reais;

Métricas de erro: Serão calculadas as principais métricas de avaliação do modelo:

RMSE (Root Mean Squared Error): Mede o erro médio entre os valores reais e os valores previstos.

MAE (Mean Absolute Error): Mede a diferença absoluta média entre as previsões e os valores reais.

$R^2$ (Coeficiente de Determinação): Mede o quão bem o modelo explica a variabilidade dos dados;

Feedback dos usuários e comparação dos preços previstos e preços reais também serão levados em consideração para avaliar a eficiência do modelo (mesmo com a troca do algoritmo por necessidades maiores).

# EXEMPLO DO ALGORITMO DE REGRESSÃO LINEAR:

```
[ ]  import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     from sklearn.model_selection import train_test_split
     from sklearn.linear_model import LinearRegression
     from sklearn.metrics import mean_absolute_error, mean_squared_error
```

```
dados = {'Distancia_km': [2, 5, 8, 10, 15, 20, 25, 30, 35],
        'Tempo_min': [5, 12, 18, 25, 30, 40, 50, 60, 70],
        'Preco_R$': [10, 20, 32, 40, 55, 70, 85, 100, 115]}

df = pd.DataFrame(dados)

# Separando as variáveis (X) e o alvo (y)
X = df[['Distancia_km', 'Tempo_min']]
y = df['Preco_R$']

# Dividindo os dados em treino (80%) e teste (20%)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Criando e treinando o modelo
modelo = LinearRegression()
modelo.fit(X_train, y_train)

# Fazendo previsões
y_pred = modelo.predict(X_test)

# Avaliação do modelo
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)

print(f'Erro Absoluto Médio (MAE): {mae:.2f}')
print(f'Erro Quadrático Médio (MSE): {mse:.2f}')

# Visualizando os resultados
plt.scatter(y_test, y_pred)
plt.xlabel('Preços Reais')
plt.ylabel('Preços Previstos')
plt.title('Regressão Linear - Previsão de Preço de Corridas')
plt.show()
```

Acima um exemplo de uma possível utilização do Algoritmo em nosso caso. Na aplicação real utilizaremos um número maior de variáveis e as bases de dados diretamente, e um estudo aprofundado sobre esse modelo deverá ser realizado para a construção do algoritmo mais preciso.

# CONCLUSÃO

Esperamos bons resultados com a utilização do algoritmo de regressão linear para prever o preço de corridas de aplicativos de transporte. Porém, alguns desafios foram identificados:

A precisão do modelo pode ser impactada por dados inconsistentes (muito presentes nas bases);

O modelo pode não capturar relações não lineares entre as variáveis;

Integrações externas, como dados de condições climáticas podem apresentar uma complexidade para ser tratadas.