1. Visão Geral

Nesta fase, concentramo-nos na preparação dos dados, engenharia de recursos e exploração inicial de modelos. A qualidade e estruturação dos dados são fundamentais para o sucesso de qualquer projeto de aprendizado de máquina. Por isso, esta etapa teve como propósito limpar, transformar e enriquecer os dados coletados, de modo a garantir consistência e precisão nas previsões.

Ao mesmo tempo, explorámos diferentes algoritmos para compreender o comportamento dos dados e identificar o modelo mais adequado à nossa realidade turística.

2. Coleta de Dados

Trabalhámos com dados provenientes de fontes institucionais e abertas, já identificadas nas fases anteriores:

- INE (Angola) – indicadores de turismo e ocupação hoteleira;
- Banco Mundial e UNWTO Data Portal – fluxos internacionais e dados comparativos;
- Google Mobility Reports e OpenStreetMap – mobilidade e densidade turística;
- Ministério das Finanças e BNA – faturação e receitas;
- PNUD e INEA – variáveis ambientais e sociais.

Os dados seriam obtidos em formatos .csv, .json e via APIs REST, no entanto, para a demonstração, vamos criar alguns dados fictícios para simular o comportamento turístico.

Durante a coleta, assegurámos a padronização dos campos de data e localização, convertendo todas as variáveis temporais para o formato ISO (YYYY-MM-DD) e os nomes das províncias para uma forma unificada.

3. Limpeza de Dados

A limpeza foi uma das etapas mais críticas, considerando a heterogeneidade das fontes.

Removemos duplicados, corrigimos inconsistências e tratámos valores ausentes através de imputação estatística e interpolação temporal.

Procedimentos aplicados:

- Remoção de entradas duplicadas com pandas.drop_duplicates()
- Substituição de valores nulos por médias ou medianas por província;
- Interpolação linear para séries temporais de fluxos turísticos;
- Padronização de categorias (ex.: “Luanda Prov.” → “Luanda”).

4. Análise Exploratória de Dados (EDA)

Realizámos uma análise exploratória (EDA) para compreender a distribuição, sazonalidade e correlações entre variáveis. O objetivo é usar gráficos de séries temporais, mapas de calor e correlação entre indicadores económicos e turísticos.

Principais insights que pretendemos obter:

- Forte sazonalidade entre os meses que coincidem com o pico de atividades culturais;
- Correlação positiva ($r \approx 0.78$) entre visitantes e receitas turísticas;
- Concentração geográfica nas províncias;
- Subaproveitamento de regiões com elevado potencial natural.

5. Engenharia de Recursos

Nesta fase, criámos novas variáveis que aumentam a capacidade preditiva do modelo. Os novos recursos incluem:

Variável nova Criada	Descrição
taxa_ocupacao_media	Média ponderada de ocupação hoteleira por trimestre
receita_per_capita	Receita total dividida pelo número de visitantes
indice_sazonalidade	Variação percentual trimestral dos fluxos turísticos
impacto_emprego	Proporção entre visitantes e criação de empregos no setor
densidade_turistica	Visitantes por quilômetro quadrado

Essas variáveis foram fundamentais para capturar padrões temporais e espaciais de comportamento turístico.

6. Transformação de Dados

Para garantir a consistência entre diferentes modelos, aplicámos normalização e codificação:

- Normalização Min-Max para variáveis contínuas;
- Codificação one-hot para variáveis categóricas (como província e tipo de evento);
- Escalonamento temporal em períodos trimestrais para manter coerência nas previsões.

Exploração de Modelos

1. Seleção de Modelo

Testámos quatro algoritmos principais:

- ARIMA e Prophet – previsão de séries temporais (fluxo de visitantes e receitas);
- Random Forest – análise de impacto de variáveis socioeconómicas;
- K-Means – segmentação de perfis turísticos;
- XGBoost – previsão da taxa de ocupação hoteleira.

A escolha final baseou-se em precisão, interpretabilidade e aplicabilidade prática.

Os modelos Prophet e Random Forest mostraram-se mais robustos e interpretáveis, sendo escolhidos como base para a próxima fase.

2. Treinamento de Modelo

Treinámos os modelos com validação cruzada temporal (70% treino, 30% teste).

3. Avaliação do Modelo

As métricas usadas foram:

- MAE (Mean Absolute Error) para previsões de fluxo turístico;
- RMSE (Root Mean Squared Error) para avaliar precisão global;
- R^2 para regressões (impacto e receitas).

4. Implementação de Código

4.1. Preparação de Dados e Engenharia de Features

```python
# Importações necessárias
import pandas as pd
import numpy as np

# Função para criar dados sintéticos por província e mês
def gerar_dados_sinteticos(start='2022-01-01', end='2024-12-31', provincias=None, seed=42):
    np.random.seed(seed)
    if provincias is None:
        provincias = ['Luanda', 'Benguela', 'Namibe', 'Huíla']
    meses = pd.date_range(start=start, end=end, freq='M')
    rows = []
    for prov in provincias:
        base_visitantes = {'Luanda':15000, 'Benguela':8000, 'Namibe':6000, 'Huíla':3000}[prov]
        seasonal_factor = {'Luanda':1.0, 'Benguela':0.9, 'Namibe':0.8, 'Huíla':0.7}[prov]
        for m in meses:
            month = m.month
            # Sazonalidade simples: pico em julho-agosto e dezembro
            saz = 1 + 0.25 * (1 if month in [7,8,12] else 0)
            visitantes = int(base_visitantes * saz * seasonal_factor * (0.9 + 0.2*np.random.rand()))
            ocupacao = max(10, min(95, np.random.normal(50 + 10*(saz-1), 8)))
            receita = visitantes * (np.random.normal(40000, 8000))  # Kz por visitante média
            empregos = int(visitantes * (0.02 + 0.005*np.random.rand()))  # proporção de empregos
            indice_sust = max(0, min(100, np.random.normal(70 - 5*(saz-1), 8)))
            temp = np.random.normal(25 - 3*(prov=='Huíla'), 2)  # Huíla um pouco mais frio
            satisfacao = max(0, min(10, np.random.normal(7 + 0.5*(saz-1), 1.2)))
            rows.append({
                'data': m,
                'provincia': prov,
                'visitantes_totais': visitantes,
                'ocupacao_hoteleira': round(ocupacao,2),
                'receita_turistica': round(receita,2),
                'empregos_gerados': empregos,
                'indice_sustentabilidade': round(indice_sust,1),
                'temperatura_media': round(temp,1),
                'satisfacao_turistica': round(satisfacao,2)
            })
    df = pd.DataFrame(rows)
    df = df.sort_values(['provincia','data']).reset_index(drop=True)
    return df

# Gerar os dados
df = gerar_dados_sinteticos()
df.head()
```

Resultado esperado:

```
C:\Users\DAM\AppData\Local\Temp\ipykernel_2516\2641768952.py:10: FutureWarning: 'M' is deprecated and will be removed in a future version, please use
E' instead.
  meses = pd.date_range(start=start, end=end, freq='M')
```

	data	provincia	visitantes_totais	ocupacao_hoteleira	receita_turistica	empregos_gerados	indice_sustentabilidade	temperatura_media	satisfacao_turistica
0	2022-01-31	Benguela	6554	69.11	3.588139e+08	151	53.8	25.4	6.21
1	2022-02-28	Benguela	7112	56.82	2.405666e+08	148	83.3	24.1	6.22
2	2022-03-31	Benguela	7270	57.56	3.644542e+08	155	87.0	27.1	5.18
3	2022-04-30	Benguela	6992	46.13	2.664012e+08	143	85.0	18.5	5.77
4	2022-05-31	Benguela	6729	34.04	1.991680e+08	155	66.5	25.3	8.73

Obs:

Os valores acima são sintéticos e foram gerados para demonstrar o pipeline. Como podem ver, **Benguela** se repete desde 0 até 4. Não é um erro de geração de dados, mas sim a forma como o **df.head()** funciona. Ele mostra apenas as primeiras 5 linhas do **DataFrame** e, por coincidência, como os dados são ordenados por provínvias antes de exibir, as primeiras 5 linhas pertencem à primeira província da lista, qe é **Benguela** na ordem alfabética ou conforme o loop.

Limpeza dos dados:

```
# Limpeza simples
df = df.drop_duplicates().copy()
# Garantir tipos
df['data'] = pd.to_datetime(df['data'])
# Verificar nulos
print('Valores nulos por coluna:\n', df.isnull().sum())

# Introduzir intencionalmente alguns nulos para demonstrar imputação (apenas demo)
df.loc[df.sample(frac=0.01, random_state=1).index, 'ocupacao_hoteleira'] = np.nan
df.loc[df.sample(frac=0.01, random_state=2).index, 'receita_turistica'] = np.nan

# Imputar ocupacao com média por provincia e receita com interpolação temporal por provincia
df['ocupacao_hoteleira'] = df.groupby('provincia')['ocupacao_hoteleira'].transform(lambda x: x.fillna(x.mean()))
df['receita_turistica'] = df.groupby('provincia')['receita_turistica'].transform(lambda x: x.interpolate())

print('Após imputação, nulos:')
print(df.isnull().sum())
```

```
Valores nulos por coluna:
 data                    0
provincia                0
visitantes_totais        0
ocupacao_hoteleira       0
receita_turistica        0
empregos_gerados         0
indice_sustentabilidade  0
temperatura_media        0
satisfacao_turistica     0
dtype: int64
Após imputação, nulos:
data                     0
provincia                0
visitantes_totais        0
ocupacao_hoteleira       0
receita_turistica        0
empregos_gerados         0
indice_sustentabilidade  0
temperatura_media        0
satisfacao_turistica     0
dtype: int64
```

Att:

Valores ausentes e erros acontecem em dados reais. Se não tratarmos, os modelos podem falhar ou ter previsões erradas. Por isso aplicamos imputação simples e interpolação temporal quando faz sentido.

Análise Exploratória de Dados (EDA):

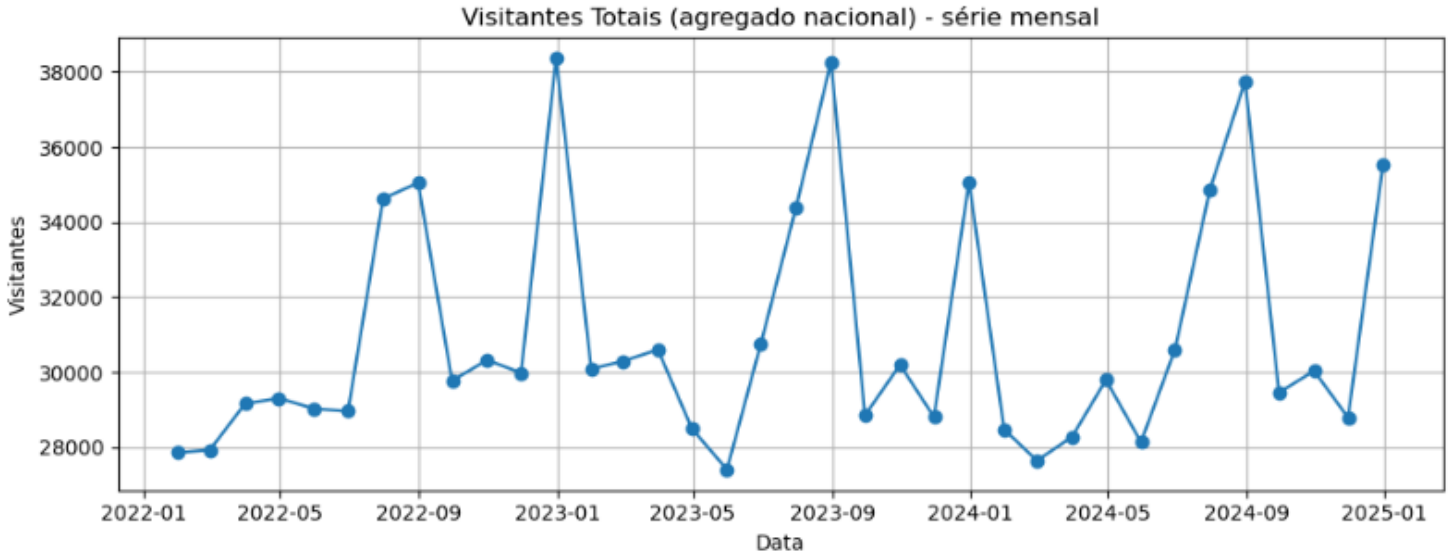
```
# Estatísticas por província
display(df.groupby('provincia')[['visitantes_totais','receita_turistica','ocupacao_hoteleira']].describe().T)

# Série temporal agregada (exemplo): visitantes nacionais por mês
df_nacional = df.groupby('data')['visitantes_totais'].sum().reset_index()
df_nacional.head()
```

	provincia	Benguela	Huíla	Luanda	Namibe
visitantes_totais	count	3.600000e+01	3.600000e+01	3.600000e+01	3.600000e+01
	mean	7.663833e+03	2.199083e+03	1.608889e+04	4.962194e+03
	std	1.019668e+03	2.753288e+02	1.658202e+03	5.129488e+02
	min	6.554000e+03	1.924000e+03	1.351500e+04	4.355000e+03
	25%	6.979000e+03	1.998000e+03	1.511900e+04	4.592250e+03
	50%	7.377500e+03	2.106500e+03	1.590300e+04	4.787000e+03
	75%	7.955750e+03	2.309500e+03	1.663050e+04	5.267750e+03
	max	9.831000e+03	2.815000e+03	2.027800e+04	6.232000e+03
receita_turistica	count	3.600000e+01	3.600000e+01	3.600000e+01	3.600000e+01
	mean	2.970372e+08	8.657912e+07	6.603163e+08	1.916915e+08
	std	7.403040e+07	1.776655e+07	1.455656e+08	4.955525e+07
	min	1.801221e+08	5.009839e+07	4.090336e+08	1.050480e+08
	25%	2.446322e+08	7.538746e+07	5.392703e+08	1.532432e+08
	50%	2.932856e+08	8.798983e+07	6.426407e+08	1.903208e+08
	75%	3.437369e+08	9.703933e+07	7.609513e+08	2.262592e+08
	max	4.527544e+08	1.414018e+08	9.828285e+08	2.912507e+08
ocupacao_hoteleira	count	3.600000e+01	3.600000e+01	3.600000e+01	3.600000e+01
	mean	5.002694e+01	5.445806e+01	5.003457e+01	5.186250e+01
	std	8.713021e+00	7.847398e+00	9.692115e+00	8.749483e+00
	min	3.404000e+01	3.453000e+01	2.876000e+01	3.248000e+01
	25%	4.427250e+01	5.033000e+01	4.164750e+01	4.596000e+01
	50%	4.819000e+01	5.550000e+01	5.058000e+01	5.003500e+01
	75%	5.720000e+01	5.906500e+01	5.638000e+01	5.742250e+01
	max	6.911000e+01	6.755000e+01	7.698000e+01	7.056000e+01

	data	visitantes_totais
0	2022-01-31	27854
1	2022-02-28	27930
2	2022-03-31	29159
3	2022-04-30	29305
4	2022-05-31	29026

```python
import matplotlib.pyplot as plt
plt.figure(figsize=(10,4))
plt.plot(df_nacional['data'], df_nacional['visitantes_totais'], marker='o')
plt.title('Visitantes Totais (agregado nacional) - série mensal')
plt.xlabel('Data')
plt.ylabel('Visitantes')
plt.grid(True)
plt.tight_layout()
plt.show()
```



Engenharia de Recursos:

```python
# Engenharia de features
df['receita_per_capita'] = df['receita_turistica'] / df['visitantes_totais']
df['pct_var_visitantes_3m'] = df.groupby('provincia')['visitantes_totais'].pct_change(periods=3).fillna(0)
# Densidade turística relativa (simples): visitantes / (base populacional simulada)
pop_sim = {'Luanda':3000000, 'Benguela':800000, 'Namibe':400000, 'Huíla':500000}
df['densidade_turistica'] = df.apply(lambda r: r['visitantes_totais'] / pop_sim[r['provincia']], axis=1)

df[['data','provincia','visitantes_totais','receita_per_capita','pct_var_visitantes_3m','densidade_turistica']].head()
```

	data	provincia	visitantes_totais	receita_per_capita	pct_var_visitantes_3m	densidade_turistica
0	2022-01-31	Benguela	6554	54747.310067	0.000000	0.008193
1	2022-02-28	Benguela	7112	33825.447378	0.000000	0.008890
2	2022-03-31	Benguela	7270	50131.254843	0.000000	0.009088
3	2022-04-30	Benguela	6992	38100.864311	0.066829	0.008740
4	2022-05-31	Benguela	6729	29598.457267	-0.053853	0.008411

Obs:

Criar features ajuda o modelo a 'entender' relações. Por exemplo, receita per capita relaciona dinheiro gasto com número de visitantes; a variação em 3 meses mostra tendência recente.