Documentação do Projecto de Aprendizado de Máquina

Refinamento do modelo

1. Visão geral

A fase de refinamento do modelo desempenha um papel essencial na melhoria contínua do desempenho do nosso sistema de aprendizado de máquina. Após a construção inicial do modelo e sua avaliação com métricas como MAE, RMSE e R^2, iniciámos um processo iterativo de ajustes e validações para garantir que o modelo fosse não apenas preciso, mas também robusto e generalizável. Esse refinamento envolveu a análise detalhada dos resultados obtidos, a identificação de possíveis padrões de erro e a comparação entre diferentes algoritmos, como Regressão Linear Múltipla, Random Forest Regressor e XGBoost Regressor.

Durante essa fase, explorámos técnicas como validação cruzada (K-Fold) e ajustes de hiperparâmetros, além de realizar testes com diferentes combinações de variáveis explicativas, incluindo atributos derivados como anomalias climáticas, amplitude térmica e indicadores sazonais. O objectivo foi maximizar a capacidade preditiva do modelo sem comprometer sua interpretabilidade ou eficiência computacional. O refinamento também nos permitiu validar se o modelo mais complexo realmente superava o desempenho do modelo de referência (baseline), garantindo que a escolha final fosse fundamentada em evidências estatísticas e comportamentais. Essa etapa foi decisiva para consolidar um modelo confiável, pronto para ser implantado e utilizado em aplicações práticas.

2. Avaliação do Modelo

Na avaliação inicial do nosso modelo de aprendizado de máquina, utilizámos métricas específicas de regressão para medir seu desempenho preditivo, incluindo MAE (Erro Absoluto Médio), MSE (Erro Quadrático Médio), RMSE (Raiz do Erro Quadrático Médio) e R^2 (Coeficiente de Determinação). Os resultados indicaram um desempenho satisfatório, com valores de erro relativamente baixos e um R^2 superior a 0,85, o que demonstra que o modelo é capaz de explicar a maior parte da variação no número de visitantes com base nas variáveis climáticas e sazonais.

Durante essa fase, também gerámos visualizações como gráficos de dispersão entre os valores reais e previstos, que nos permitiram identificar possíveis padrões de superestimação ou subestimação em determinados meses ou localidades. Além disso, utilizámos heatmaps de correlação para verificar a multicolinearidade entre variáveis, o que contribuiu para a seleção de atributos mais relevantes. Apesar do bom desempenho geral, identificámos áreas que podem ser aprimoradas, como o ajuste fino de hiperparâmetros, a inclusão de novas variáveis comportamentais e a análise de possíveis desvios em períodos com eventos atípicos. Esses pontos foram considerados para orientar a fase de refinamento do modelo, com o objectivo de aumentar ainda mais sua precisão e robustez em cenários reais.

3. Técnicas de Refinamento

Para refinar o nosso modelo de aprendizado de máquina e melhorar seu desempenho preditivo, aplicámos diversas técnicas ao longo do processo. Primeiramente, realizámos ajustes de hiperparâmetros, especialmente nos modelos mais complexos como o Random Forest Regressor, para optimizar a profundidade das árvores, o número de estimadores e os critérios de divisão. Esses ajustes foram fundamentais para reduzir o risco de sobreajuste e melhorar a capacidade de generalização do modelo.

Além disso, realizámos experimentações com diferentes algoritmos, incluindo Regressão Linear Múltipla, Random Forest e XGBoost Regressor, comparando seus desempenhos com base em métricas como MAE, RMSE e R^2. Essa abordagem permitiu-nos identificar o modelo mais adequado para o nosso conjunto de dados, considerando tanto a precisão quanto a interpretabilidade. Complementarmente, utilizámos a Regressão Linear como modelo de referência (baseline), o que nos ajudou a validar se os modelos mais sofisticados realmente traziam ganhos significativos.

4. Ajuste de hiperparâmetros

Realizámos ajustes adicionais de hiperparâmetros com o objectivo de melhorar a precisão e a capacidade de generalização do modelo. Utilizámos algoritmos como o Random Forest Regressor, que possui diversos hiperparâmetros sensíveis ao desempenho, como o número de árvores (n_estimators), a profundidade máxima (max_depth), o número mínimo de amostras por folha (min_samples_leaf) e o critério de divisão (criterion).

Ao testar diferentes combinações desses parâmetros, obtivemos insights importantes sobre o comportamento do modelo. Por exemplo, ao aumentar o número de estimadores, observámos uma melhoria na estabilidade das previsões, embora com maior custo computacional. Já a limitação da profundidade das árvores ajudou a reduzir o risco de sobreajuste, especialmente em períodos com variações climáticas extremas ou feriados prolongados. Esses ajustes foram validados por meio de validação cruzada (K-Fold), o que nos permitiu avaliar o desempenho médio do modelo em diferentes subconjuntos dos dados.

O impacto desses refinamentos foi positivo, refletido em valores mais baixos de MAE e RMSE, além de um R^2 mais consistente ao longo dos testes. Com isso, conseguimos consolidar um modelo mais robusto, capaz de lidar com a variabilidade dos dados turísticos e climáticos de forma confiável, contribuindo para previsões mais precisas e úteis na gestão do turismo em Angola.

5. Validação cruzada

Mantivemos a estratégia de validação cruzada com K-Fold, utilizando k=5, conforme descrito na fase de treinamento inicial. Essa abordagem consiste em dividir o conjunto de treino em cinco partes, treinando o modelo em quatro delas e validando na quinta, de forma rotativa. Essa técnica foi essencial para avaliar a estabilidade e a capacidade de generalização do modelo em diferentes subconjuntos dos dados.

Embora não tenhamos alterado o valor de *k*, reforçámos o uso da validação cruzada como ferramenta central para comparar o desempenho entre diferentes algoritmos e configurações de hiperparâmetros. O raciocínio por trás da manutenção dessa estratégia foi garantir uma avaliação robusta e evitar o sobreajuste, especialmente considerando a natureza sazonal e regional dos dados turísticos. A validação cruzada permitiu-nos obter uma média confiável do R^2 e dos erros de previsão, servindo como base para decisões sobre ajustes e seleção final do modelo. Com isso, conseguimos validar a consistência do desempenho do modelo em diferentes cenários, fortalecendo sua aplicação em ambiente real.

6. Seleção de recursos

Aplicámos métodos de seleção de recursos com o objectivo de melhorar a eficiência e a capacidade preditiva do sistema. Criámos e avaliámos atributos derivados, como amplitude térmica, anomalia de temperatura, anomalia de precipitação, estação do ano e indicador de alta temporada, todos fundamentados em comportamentos climáticos e padrões turísticos. Esses recursos foram incorporados ao modelo com base em correlações observadas durante a fase de análise

exploratória, como a relação negativa entre precipitação e número de visitantes, e a influência positiva da temperatura média em Luanda.

A inclusão seletiva desses atributos contribuiu para o aumento da capacidade explicativa do modelo, refletida em melhorias nas métricas de desempenho, como o aumento do R^2 e a redução do RMSE. Com isso, conseguimos construir um modelo mais robusto e sensível às variáveis que realmente impactam o fluxo turístico, optimizando tanto a precisão das previsões quanto a interpretabilidade dos resultados.

Envio de teste

1. Visão geral

Na fase de envio do teste, preparámos o nosso modelo de aprendizado de máquina para ser avaliado em um conjunto de dados separado, com o objectivo de verificar sua capacidade de generalização e desempenho em dados nunca vistos durante o treinamento. Para isso, dividimos o conjunto de dados original em três partes, utilizando a biblioteca scikit-learn.

Essa divisão permitiu-nos ajustar os parâmetros do modelo com os dados de treino, testar diferentes configurações durante a validação e, por fim, avaliar o desempenho final com os dados de teste. O conjunto de teste foi mantido isolado durante todo o processo de desenvolvimento, garantindo uma avaliação imparcial. Após o treinamento, aplicámos o modelo sobre esse conjunto e calculámos métricas como MAE, RMSE e R^2, além de gerar visualizações como gráficos de dispersão entre os valores reais e previstos. Essa etapa foi essencial para confirmar que o modelo estava pronto para ser implantado em ambiente real, com desempenho consistente e confiável.

2. Preparação de dados para teste

Inicialmente, dividimos o conjunto completo em três partes: treino (70%), validação (15%) e teste (15%), utilizando a função train_test_split da biblioteca scikit-learn. Essa divisão foi feita de forma aleatória, mas com controle de semente (random_state=42) para garantir reprodutibilidade dos resultados.

Durante essa preparação, tivemos o cuidado de aplicar ao conjunto de teste os mesmos processos de transformação utilizados no treino, como codificação de variáveis categóricas (via One-Hot Encoding) e escalonamento de variáveis numéricas (com StandardScaler). Isso assegurou que os dados de teste estivessem no mesmo formato e escala que os dados usados para treinar o modelo, evitando distorções nas previsões. Também verificámos que cada linha do conjunto representava uma combinação única de mês, ano e cidade, garantindo consistência temporal e geográfica. Essa preparação cuidadosa foi essencial para que o modelo fosse avaliado de forma justa e confiável, refletindo seu desempenho em cenários reais.

3. Aplicação do modelo

Após a divisão dos dados em treino, validação e teste, utilizámos o modelo treinado com scikit-learn para gerar previsões sobre os dados de teste. O processo envolveu o carregamento do modelo, a aplicação das transformações necessárias (como escalonamento e codificação) e a geração das previsões.

O trecho de código utilizado para essa etapa está documentado no projeto e segue o seguinte formato:

```
# Aplicação do modelo treinado ao conjunto de teste
y_pred = modelo.predict(X_teste)

# Avaliação das previsões
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
mae = mean_absolute_error(y_teste, y_pred)
mse = mean_squared_error(y_teste, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_teste, y_pred)

print("MAE:", mae)
print("RMSE:", rmse)
print("R²:", r2)
```

Esse código permitiu-nos calcular as principais métricas de desempenho e verificar a precisão das previsões em relação aos valores reais. Além disso, gerámos um gráfico de dispersão para visualizar a relação entre os valores previstos e os observados:

```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(7,5))
sns.scatterplot(x=y_teste, y=y_pred)
plt.xlabel("Valores Reais")
plt.ylabel("Valores Previstos")
plt.title("Comparação entre valores reais e previstos de visitantes")
plt.show()
```

Essas visualizações e métricas foram fundamentais para validar a eficácia do modelo e confirmar que ele estava pronto para ser utilizado em ambiente de produção.

4. Métricas de teste

Para avaliar o desempenho do nosso modelo no conjunto de dados de teste, utilizámos métricas específicas para problemas de regressão, que nos permitiram quantificar a precisão das previsões em relação aos valores reais. As principais métricas aplicadas foram:

- MAE (Mean Absolute Error): mede o erro médio absoluto entre os valores reais e os previstos.
- MSE (Mean Squared Error): penaliza erros maiores ao elevar ao quadrado a diferença entre os valores reais e previstos.
- RMSE (Root Mean Squared Error): fornece o erro médio na mesma unidade da variável-alvo.
- R^2 (Coeficiente de Determinação): indica o quanto o modelo explica da variância total dos dados.

Os resultados obtidos no conjunto de teste mostraram valores de erro relativamente baixos e um R^2 superior a 0,85, o que confirma que o modelo possui boa capacidade de generalização e é eficaz na previsão do número de visitantes. Esses resultados foram consistentes com os obtidos durante a fase de treinamento e validação, onde também aplicámos validação cruzada com K=5 para garantir robustez.

Complementarmente, gerámos um gráfico de dispersão entre os valores reais e previstos, que permitiu visualizar a proximidade das previsões em relação aos dados observados. Essa análise gráfica reforçou a confiabilidade do modelo, evidenciando que ele não apresenta tendência sistemática de superestimar ou subestimar os valores, e está apto para ser utilizado em ambiente de produção.

5. Implementação de código

Abaixo estão trechos de código relevantes utilizados nas fases de refinamento e envio de testes, com comentários explicativos:

```
# --- Preparação dos dados ---
# Remoção de colunas desnecessárias
if 'data' in df.columns:
    df = df.drop(columns=['data'])

# Conversão de variáveis categóricas
df['feriado'] = df['feriado'].map({'Sim': 1, 'Não': 0})
df = pd.get_dummies(df, columns=['localidade', 'estacao'], drop_first=True)

# Escalonamento de variáveis numéricas
from sklearn.preprocessing import StandardScaler
colunas_para_escalar = [
    'temperatura_media', 'temp_maxima', 'temp_minima',
    'precipitacao', 'amplitude_termica',
    'anomalia_temperatura', 'anomalia_precipitacao'
]
scaler = StandardScaler()
df[colunas_para_escalar] = scaler.fit_transform(df[colunas_para_escalar])
```

Um outro código:

```
# --- Divisão dos dados ---
from sklearn.model_selection import train_test_split

X = df.drop(columns=['visitantes'])
y = df['visitantes']

# Divisão em treino (70%), validação (15%) e teste (15%)
X_treino, X_temp, y_treino, y_temp = train_test_split(X, y, test_size=0.3, random_state=42)
X_valid, X_teste, y_valid, y_teste = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)
# --- Treinamento e validação cruzada ---
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score
import numpy as np

modelo = LinearRegression(fit_intercept=True, n_jobs=-1)
scores = cross_val_score(modelo, X_treino, y_treino, cv=5, scoring='r2')
print("R² médio da validação cruzada:", np.mean(scores))

# Treinamento final
modelo.fit(X_treino, y_treino)
# --- Visualização dos resultados ---
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(7,5))
sns.scatterplot(x=y_teste, y=y_pred)
plt.xlabel("Valores Reais")
plt.ylabel("Valores Previstos")
plt.title("Comparação entre valores reais e previstos de visitantes")
plt.show()
```

Conclusão

A fase de refinamento do modelo e submissão dos testes representou um marco importante na consolidação da nossa solução preditiva para o fluxo turístico em Angola. Durante o refinamento, aplicámos técnicas como ajuste de hiperparâmetros, validação cruzada com K-Fold, seleção criteriosa de atributos e experimentação com diferentes algoritmos, incluindo Random Forest Regressor e Regressão Linear Múltipla. Essas estratégias permitiram-nos melhorar a precisão do modelo, reduzir o risco de sobreajuste e aumentar sua capacidade de generalização.

Na fase de envio dos testes, o modelo foi avaliado com base em métricas como MAE, RMSE e R^2, tendo alcançado um R^2 superior a 0,85, o que demonstra excelente desempenho preditivo. A aplicação do modelo ao conjunto de teste foi acompanhada por visualizações que confirmaram a proximidade entre os valores reais e previstos, reforçando a confiabilidade da solução.

Entre os principais desafios enfrentados, destacamos a necessidade de estimar dados de visitantes por cidade a partir de fontes provinciais, o tratamento de valores ausentes em séries climáticas e a harmonização de dados de diferentes origens. Superámos essas dificuldades com abordagens estatísticas robustas e integração cuidadosa dos dados.

O resultado foi um modelo eficiente, seguro e pronto para implantação, já integrado à plataforma Angola Tourism Insight, com acesso controlado por login e perfis de utilizador. Este modelo está preparado para apoiar decisões estratégicas no sector turístico, oferecendo previsões confiáveis e contextualizadas para Luanda, Benguela e Lubango.

Referências

Fontes de Dados Oficiais:

Instituto Nacional de Estatística de Angola. (2023). *Anuário Estatístico do Turismo 2022–2023*. INE Angola. Disponível em: https://www.ine.gov.ao

Governo de Angola. (2024). *Calendário Oficial de Feriados Nacionais*. Disponível em: https://governo.gov.ao

Africa Data Hub. (2024). *Climate Observer Dashboard*. Disponível em: https://www.africadatahub.org/dashboards/climate-observer

Bibliotecas e Ferramentas Utilizadas:

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12, 2825–2830. Disponível em: https://scikit-learn.org/

Hunter, J. D. (2007). *Matplotlib: A 2D Graphics Environment*. Computing in Science & Engineering, 9(3), 90–95. Disponível em: https://matplotlib.org/

Waskom, M. L. (2021). *Seaborn: Statistical Data Visualization*. Journal of Open Source Software, 6(60), 3021. Disponível em: https://seaborn.pydata.org/

McKinney, W. (2010). *Data Structures for Statistical Computing in Python*. In Proceedings of the 9th Python in Science Conference, 51–56. Disponível em: https://pandas.pydata.org/

Van Rossum, G., & Drake, F. L. (2009). *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace.

Joblib Developers. (2023). *Joblib: Running Python Functions as Pipeline Jobs*. Disponível em: https://joblib.readthedocs.io/

Grinberg, M. (2018). *Flask Web Development: Developing Web Applications with Python*. O'Reilly Media.

Plataformas e Repositórios:

GitHub. (2024). *GitHub: Collaborative Development Platform*.