INSTITUTO FEDERAL
Catarinense
Campus Rio do Sul

Relatório: Previsão do Nível do Rio do Sul com Regressão Linear

Magdiel Prestes Rodrigues

magdielprestes@gmail.com,

Rio do Sul, SC
Julho, 2025.

Introdução

Este trabalho utiliza um dataset real contendo informações de níveis de rios e chuvas para prever o nível do rio em Rio do Sul por meio de regressão linear. O objetivo é seguir as etapas obrigatórias da disciplina, incluindo análise exploratória, pré-processamento, treinamento do modelo, salvamento e implementação de uma aplicação interativa. A linguagem utilizada foi Python, no ambiente Jupyter Notebook, e a entrega está em formato PDF com link para o repositório GitHub.

1. Leitura e Análise Inicial dos Dados

O dataset foi carregado diretamente do arquivo dados_rio_e_chuva.xlsx usando a biblioteca pandas. Para garantir a interpretação dos dados numéricos, foi implementada uma conversão automática de vírgulas para pontos:

```
for col in dados.columns:
    if dados[col].dtype == 'object':
        try:
            dados[col] = dados[col].str.replace(',', '.').astype(float)
        except (ValueError, AttributeError):
            pass
```

Resultados da análise inicial:

Primeiras linhas do dataset:

	NivelRiodoSul	NivelItuporanga	ChuvaItuporanaga	NivelTaio	ChuvaTaio
0	170	30	3.0	98	0.0
1	170	39	12.0	98	0.0
2	169	59	19.0	98	0.0
3	169	82	5.0	98	0.0
4	169	27	1.0	97	0.0

- **Shape do dataset:** (65408, 5).
- **Nomes das colunas:** ['NivelRiodoSul', 'NivelItuporanga', 'ChuvaItuporanaga', 'NivelTaio', 'ChuvaTaio'].
- **Tipos de dados:** Todos confirmados como numéricos após conversão.
- **Valores faltantes:** Nenhum registrado (0 por coluna).
- **Shape após remover duplicatas:** (40277, 5).

Análise Exploratória

Foram gerados dois gráficos principais para compreender os dados. O primeiro foi um histograma da variável alvo (Nível do Rio do Sul), gerado com sns.histplot(dados['NivelRiodoSul'], bins=30, kde=True), que mostra a distribuição dos valores do nível do rio. O segundo gráfico foi um scatter plot (Nível do Rio do Sul vs. Chuva Ituporanga), criado com sns.scatterplot(x='ChuvaItuporanaga', y='NivelRiodoSul', data=dados), que visualiza a relação entre precipitação e nível do rio.

2. Pré-processamento

Seleção de Features

As features foram selecionadas com base na correlação absoluta com a variável alvo (NivelRiodoSul):

```
correlacoes = dados.corr()['NivelRiodoSul'].abs().sort_values(ascending=False)
features_selecionadas = correlacoes[1:5].index.tolist()
```

Normalização dos Dados de Nível

Uma etapa importante foi a normalização dos dados de nível (dividindo por 100) para compatibilizar as escalas:

```
for feature in features_selecionadas + ['NivelRiodoSul']:
    if feature.startswith('Nivel'):
        dados[feature] = dados[feature] / 100
```

Tratamento de Valores Faltantes e Normalização

Os valores faltantes foram preenchidos com a média de cada coluna, as variáveis de entrada foram normalizadas usando MinMaxScaler, e o dataset foi dividido em 70% para treino e 30% para teste (random_state=42)..

3. Treinamento do Modelo

Otimização de Combinações de Features

O código implementa uma busca exaustiva por todas as combinações possíveis de features:

```
from itertools import combinations
```

```
best_score = -np.inf
best_features = []
best_scaler = None

for r in range(1, len(features_selecionadas) + 1):
    for combo in combinations(features_selecionadas, r):
        # Treina modelo para cada combinação
        # Seleciona a melhor baseada no R²
```

Modelo Final

O modelo de regressão linear foi treinado com a melhor combinação de features encontrada:

```
model = LinearRegression()
model.fit(X_train_scaled_best, y_train)
```

4. Avaliação do Modelo

As métricas foram calculadas com base no conjunto de teste:

```
y_pred = model.predict(X_test_scaled_best)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

Métricas do modelo final:

- **RMSE:** [Valor obtido na execução].
- **MAE:** [Valor obtido na execução].
- **R²:** [Valor obtido na execução].

5. Salvamento do Modelo

O modelo treinado, scaler e features foram salvos usando joblib:

```
best_features_no_accents = [feature.encode().decode('ascii', 'ignore') for feature in best_features]
joblib.dump(model, 'modelo_regressao_linear.pkl')
```

```
joblib.dump(best_scaler, 'scaler.pkl')
joblib.dump(best_features_no_accents, 'features.pkl')
```

6. Aplicação

Função de Previsão

A função de previsão foi implementada com carregamento adequado dos componentes salvos:

```
def prever_nivel_rio(dados_entrada):
    model_carregado = joblib.load('modelo_regressao_linear.pkl')
    scaler_carregado = joblib.load('scaler.pkl')
    features_carregadas = joblib.load('features.pkl')

    # Organiza entrada conforme features treinadas
    ordered_input = {k: dados_entrada.get(k, 0.0) for k in sorted(features_carregadas)}

    # Aplica mesma transformação do treinamento
    entrada = pd.DataFrame([ordered_input], columns=features_carregadas)
    entrada_scaled = scaler_carregado.transform(entrada)

    previsao = model_carregado.predict(entrada_scaled)[0]
    return previsao
```

Exemplo de Uso

```
exemplo_entrada = {'NivelItuporanga': 4.0, 'NivelTaio': 6.0}
previsao = prever_nivel_rio(exemplo_entrada)
print(f"Nível previsto do rio: {previsao:.2f} metros")
```

Aspectos Técnicos Importantes

O desenvolvimento do modelo abordou diversos aspectos técnicos importantes para garantir sua funcionalidade e aplicação. Foi implementado um tratamento de encoding com remoção de acentos para compatibilidade, evitando problemas de codificação no salvamento e carregamento do modelo. A metodologia incluiu uma busca exaustiva testando todas as combinações possíveis de features, garantindo a seleção da melhor configuração baseada no coeficiente de determinação R^2.

Para manter a consistência de escala, foi aplicada a mesma transformação tanto no treinamento quanto na predição, utilizando o scaler salvo durante o processo de treinamento. O modelo também demonstra bom funcionamento através do tratamento

adequado de valores ausentes com defaults, preenchendo automaticamente com zero quando uma feature não é fornecida na entrada da função de previsão.

Conclusão

O modelo de regressão linear implementado utiliza uma abordagem sistemática para seleção de features e otimização de performance. A implementação garante reprodutibilidade através do salvamento adequado do modelo, scaler e features utilizadas. A função de previsão mantém consistência com o processo de treinamento, permitindo aplicação prática do modelo desenvolvido.

Link do GitHub: https://github.com/MagdielPr/IA-TrabalhoFinal

Observação: Os valores específicos das métricas devem ser preenchidos após a execução completa do código no ambiente Jupyter Notebook.