Relatório Wesley Antonio.

RGA: 202011722024

Descrição do Conjunto de Dados:

O conjunto de dados foi gerado utilizando o script gen.py e tem como objetivo criar um cenário realista para análise preditiva por meio de modelos de regressão. A seguir, são apresentadas as principais características do dataset.

1. **Amostragem e Tamanho**

    a. O dataset contém 1.000.000 de amostras.
    b. A geração dos dados utilizou uma semente fixa (seed=42), garantindo reprodutibilidade dos resultados.

2. **Estrutura das Features**

    a. O conjunto de dados é composto por 30 features distribuídas em três grupos distintos:
    b. Features Base (10 variáveis):
    c. Geradas a partir de uma distribuição normal, essas features representam a base do conjunto e servem de referência para a criação das demais variáveis.
    d. Features Derivadas (10 variáveis)
    e. Obtidas por meio de transformações não-lineares e operações aritméticas aplicadas às features base (como multiplicação, funções trigonométricas, logaritmo e tangente hiperbólica), essas features adicionam variabilidade e complexidade ao conjunto.
    f. Features Irrelevantes (10 variáveis):
    g. Geradas a partir de uma distribuição uniforme no intervalo de -10 a 10, essas variáveis não possuem relação direta com a variável alvo. Elas podem ser úteis para testar a robustez dos modelos em identificar as features realmente relevantes

3. **Variável Alvo (Target)**

    a. A variável target foi calculada como uma combinação linear ponderada de determinadas features base e derivadas, somada a um termo de ruído proveniente de uma distribuição normal. Essa abordagem adiciona uma variabilidade que simula condições reais encontradas em problemas de regressão.

4. **Formato do Arquivo CSV**

    a. Primeira Linha: Contém as etiquetas que identificam o tipo de cada coluna (base, derivada, irrelevante e target), facilitando a interpretação dos dados.
    b. Segunda Linha: Apresenta os nomes reais das colunas.
    c. Linhas Subsequentes: Contêm os registros individuais do dataset.

Este conjunto de dados é especialmente adequado para a validação de técnicas de ensemble e outros métodos de regressão, pois oferece um volume expressivo de dados. Isso permite realizar comparativos aprofundados e avaliar tanto a performance quanto a robustez dos modelos.

Exemplo dos dados:

Dimensão do dataset: (1000000, 31)
Visualização das primeiras linhas:

```
        f0        f1        f2        f3        f4        f5        f6  \
0  0.496714 -0.138264  0.647689  1.523030 -0.234153 -0.234137  1.579213
1 -0.463418 -0.465730  0.241962 -1.913280 -1.724918 -0.562288 -1.012831
2  1.465649 -0.225776  0.067528 -1.424748 -0.544383  0.110923 -1.150994
3 -0.601707  1.852278 -0.013497 -1.057711  0.822545 -1.220844  0.208864
4  0.738467  0.171368 -0.115648 -0.301104 -1.478522 -0.719844 -0.460639

        f7        f8        f9  ...       f21       f22       f23       f24  \
0  0.767435 -0.469474  0.542560  ... -4.597700  1.543534 -3.960569 -4.507103
1  0.314247 -0.908024 -1.412304  ...  4.077747  1.660002  0.552908  4.364806
2  0.375698 -0.600639 -0.291694  ... -4.505449  6.792826 -6.604960 -6.806906
3 -1.959670 -1.328186  0.196861  ... -5.278037 -0.055900 -3.260408  4.047104
4  1.057122  0.343618 -1.763040  ...  2.210531  2.621112  1.165708 -9.792207

       f25       f26       f27       f28       f29    target
0  3.012064  7.135700  9.841085  7.736855 -8.380687 -0.049920
1 -6.649625  2.114922 -0.484172 -0.792471 -8.861462  0.578067
2  5.717368 -9.379581  1.617772  4.692927 -0.392823  1.334354
3 -1.073594  7.951624  7.188665  1.441897  4.015262 -2.099248
4 -1.053429  8.768383  1.609174 -1.416970 -8.997578 -2.170465
```

Pipeline de pré-processamento:

1. **Tratamento de Valores Faltantes**

    a. Inicialmente, é verificado se existem valores faltantes em cada coluna. Caso sejam encontrados, eles podem ser imputados ou removidos. Na implementação atual não há valores faltantes então não é necessário fazer nenhum tratamento. Exemplo do código implementado para a validação de valores nulos:

```
print("\nValores faltantes por coluna:")
print(df.isnull().sum())
```

2. **Separação das Features e Variável Alvo**

    a. A coluna target é separada das demais variáveis para que as features possam ser normalizadas independentemente. Código usado para a separação das colunas :

```
features = df.drop('target', axis=1)
target = df['target']
```

3. **Normalização com StandardScaler**

    a. Para tratar a disparidade das escalas entre as variáveis, é utilizada a normalização com o StandardScaler. Essa etapa transforma as features para que tenham média zero e desvio padrão igual a um. Após a normalização, os dados são convertidos novamente para um DataFrame, mantendo a identificação das colunas originais. Codigo usado abaixo:

```
scaler = StandardScaler()
features_normalized = scaler.fit_transform(features)

features_normalized = pd.DataFrame(features_normalized,
columns=features.columns)
```

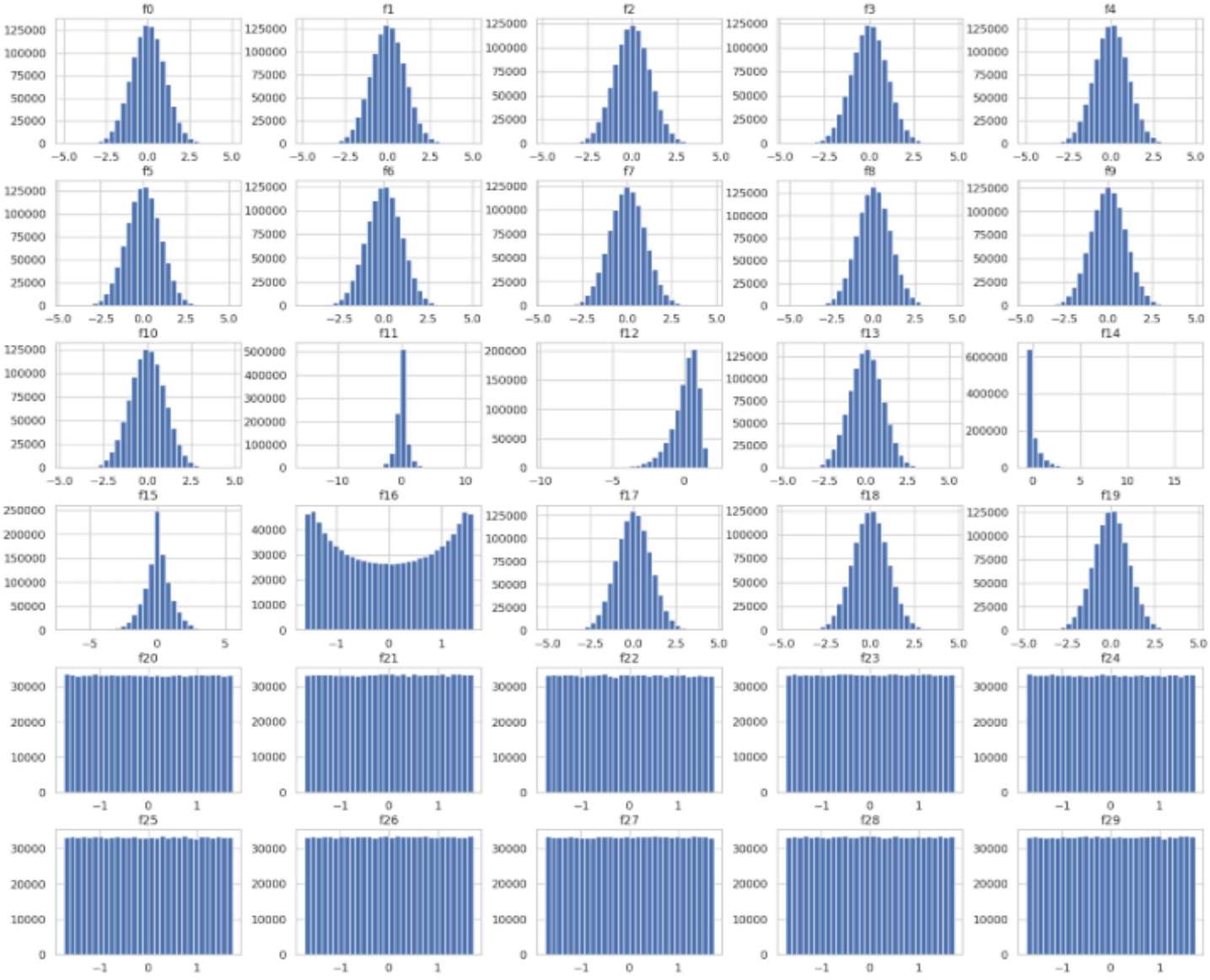
4. **Recomposição do DataFrame**

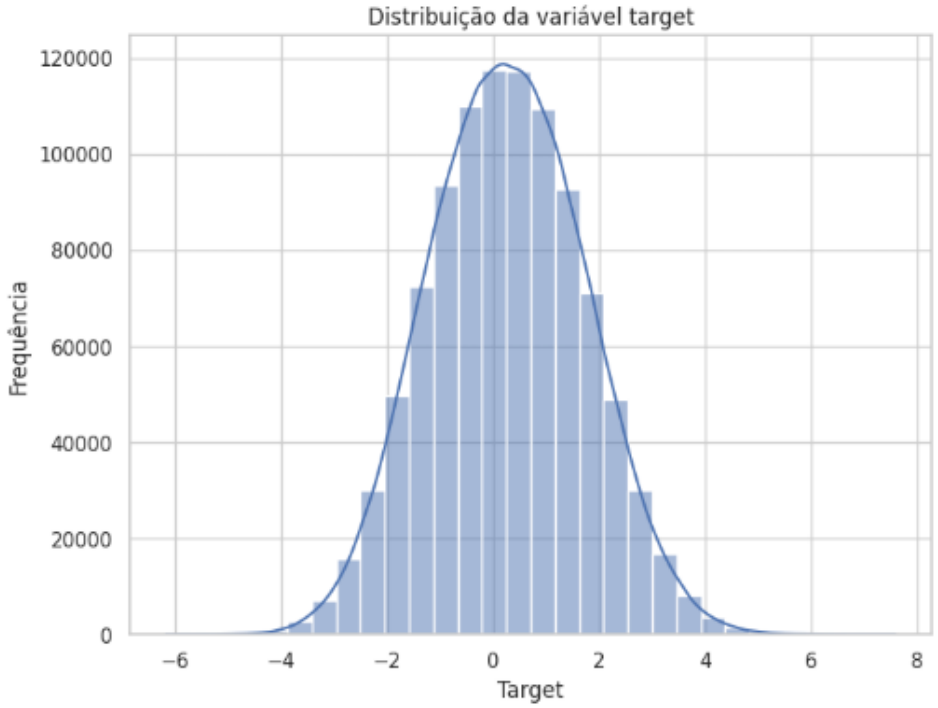
    a. Finalmente, a coluna target é reconectada ao conjunto das features normalizadas, formando um novo DataFrame que será utilizado para o treinamento e avaliação dos modelos:

```
df_n = pd.concat([features_normalized, target], axis=1)
```

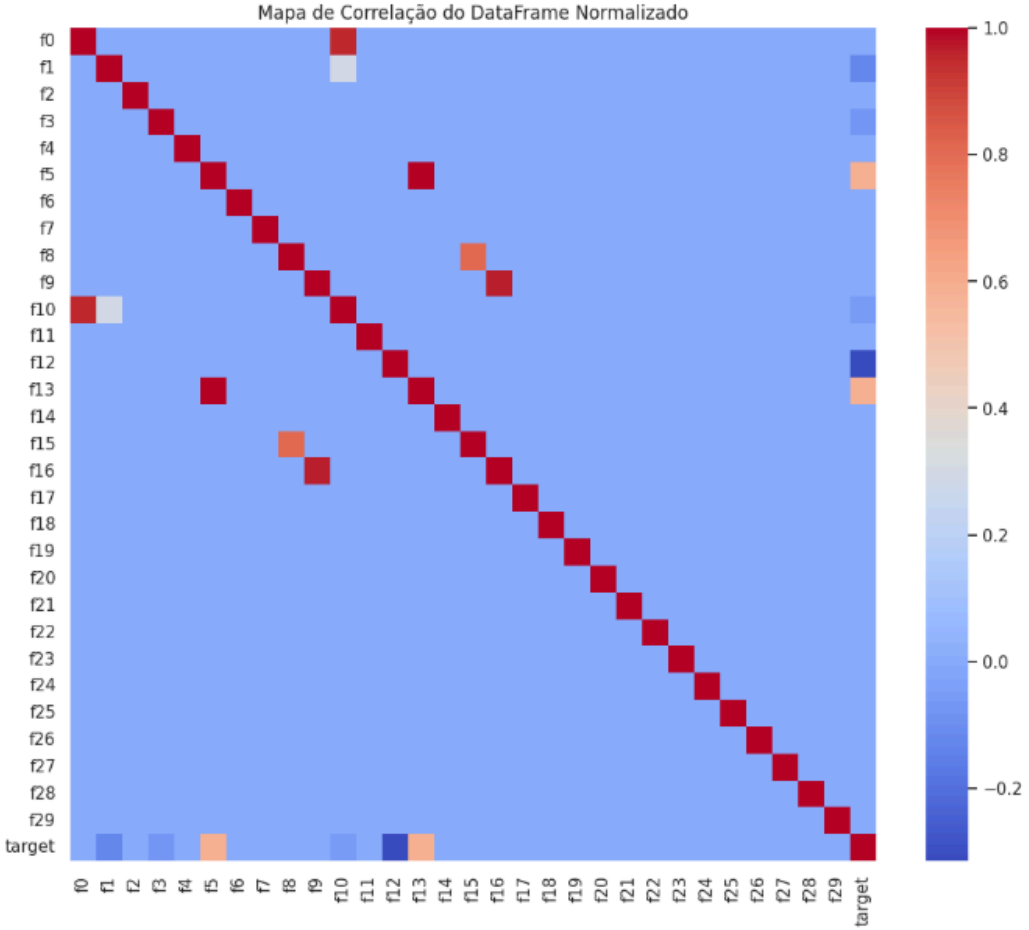
Exploração:

Histogramas das variáveis:





Heat map da correlação:



Estratégia de construção dos modelos:

A estratégia de construção dos modelos foi estruturada em duas etapas principais:

1. **Modelos Base:**

    a. Foi utilizada uma abordagem com algoritmos individuais (base learners) como Linear Regression, SVR e KNN.
    b. Cada modelo foi treinado separadamente utilizando os dados normalizados.
    c. Métricas como RMSE, MAE, R^2 e MAPE foram calculadas para avaliar o desempenho individual, e tempos de treino e predição foram registrados.

2. **Modelos Ensemble:**

    a. Foram implementadas técnicas de ensemble que combinam as previsões dos modelos base para potencialmente melhorar a performance preditiva.
    b. Dois principais métodos foram adotados:
        i. Bagging: Utilizando o Random Forest Regressor, que agrega as previsões de várias árvores construídas sobre amostras aleatórias do conjunto de treinamento.
        ii. Boosting: Através do Gradient Boosting Regressor, que constrói os modelos sequencialmente, corrigindo os erros dos modelos anteriores.
    c. Além disso, foi implementado um ensemble customizado que calcula a média simples das previsões dos modelos base, buscando avaliar se uma fusão direta das saídas melhora os resultados.

Métricas computadas e análise crítica:

Métricas Computadas:

- RMSE (Root Mean Squared Error).
- MAE (Mean Absolute Error).
- R^2 (Coeficiente de Determinação).
- MAPE (Mean Absolute Percentage Error).

Saídas:

Base: LinearRegression | RMSE: 1.0641 | MAE: 0.8426 | R2: 0.4734 | MAPE: 385.24%
Base: SVR | RMSE: 1.0341 | MAE: 0.8243 | R2: 0.5027 | MAPE: 439.16%
Base: KNN | RMSE: 1.1932 | MAE: 0.9512 | R2: 0.3380 | MAPE: 420.33%

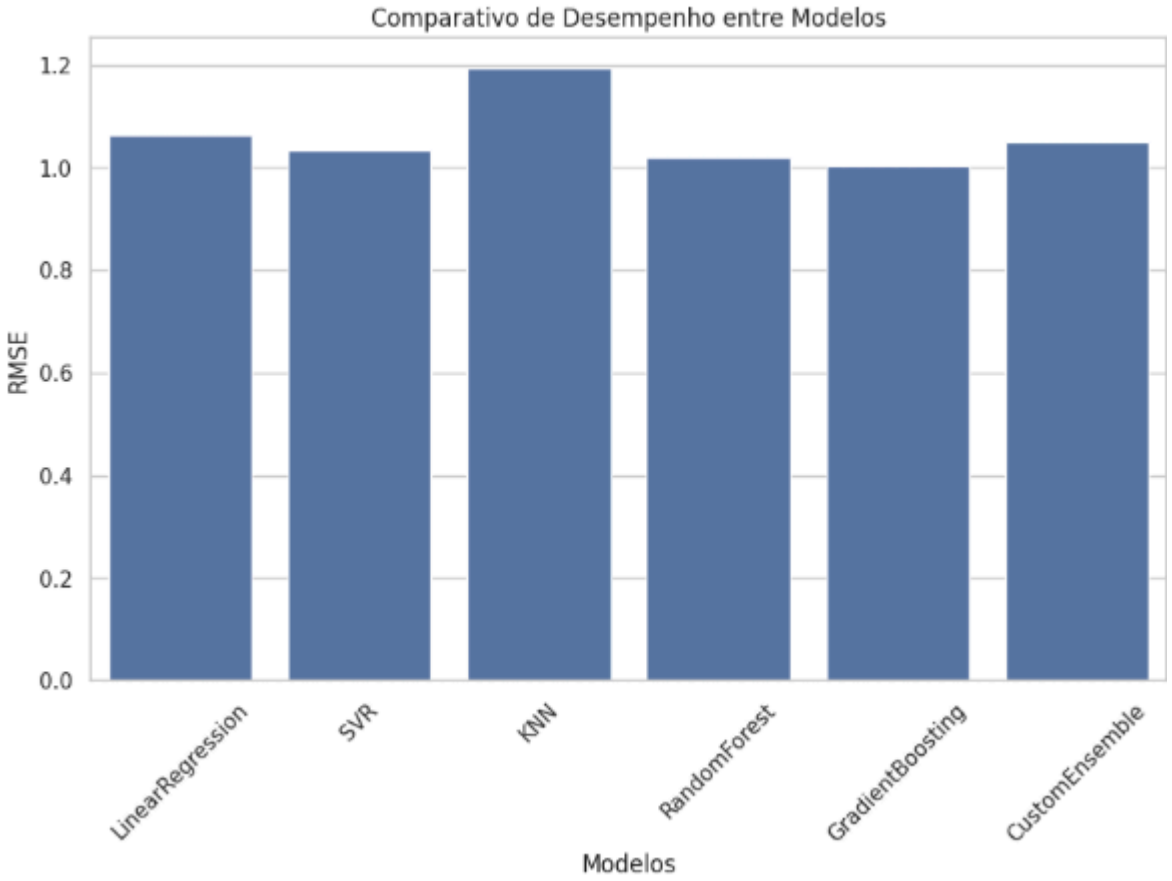
Ensemble - RandomForest | RMSE: 1.0188 | MAE: 0.8121 | R2: 0.5173 | MAPE: 439.39%
Ensemble - GradientBoosting | RMSE: 1.0036 | MAE: 0.8001 | R2: 0.5316 | MAPE: 424.77%
Ensemble - Custom (Média) | RMSE: 1.0494 | MAE: 0.8361 | R2: 0.4879 | MAPE: 377.49%

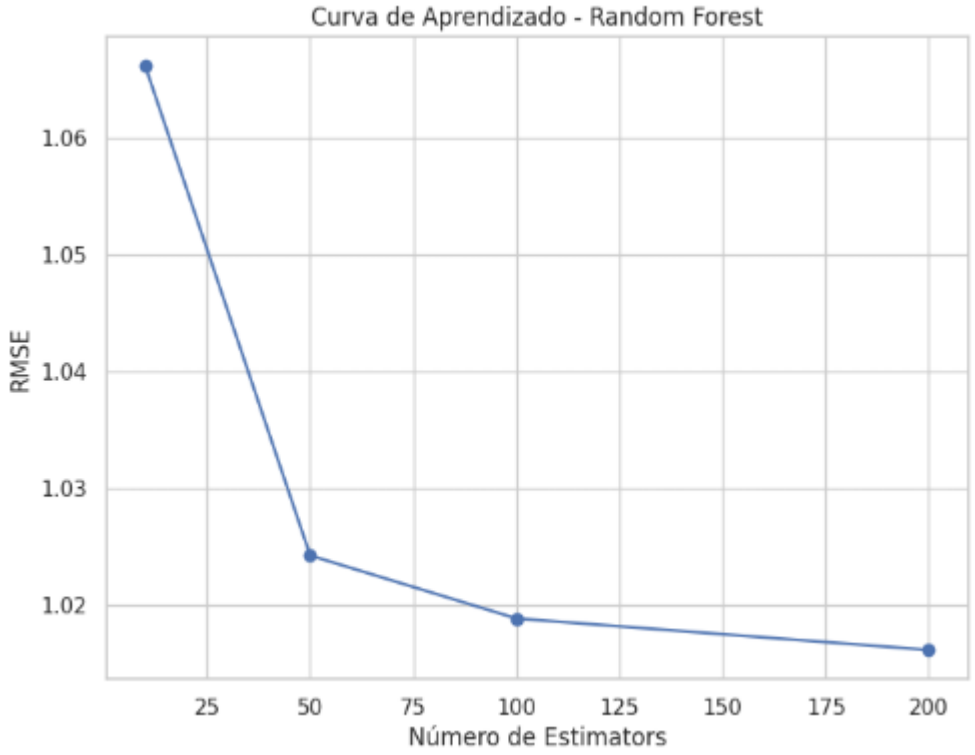
Análise Crítica:

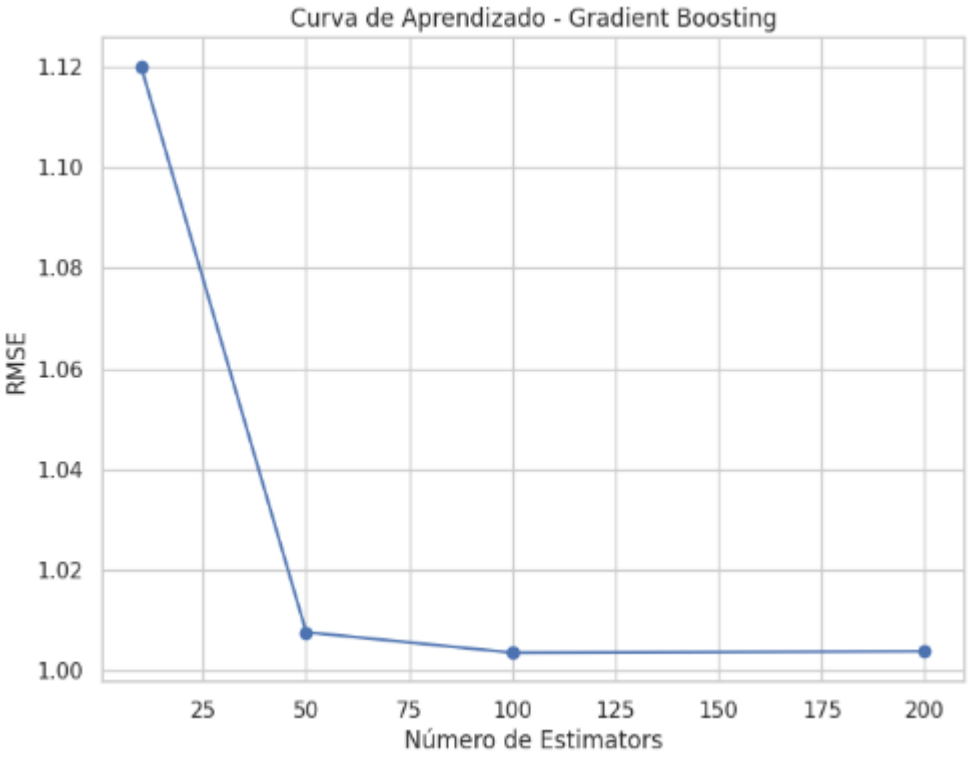
Modelos ensemble, como o Random Forest (RMSE: 1.0188, MAE: 0.8121, R^2: 0.5173, MAPE: 439.39%) e o Gradient Boosting (RMSE: 1.0036, MAE: 0.8001, R^2: 0.5316, MAPE: 424.77%), apresentaram desempenho superior em comparação com modelos base como a Regressão Linear (RMSE: 1.0641, MAE: 0.8426, R^2: 0.4734, MAPE: 385.24%) e o SVR (RMSE: 1.0341, MAE: 0.8243, R^2: 0.5027, MAPE: 439.16%). O KNN teve o pior desempenho (RMSE: 1.1932, MAE: 0.9512, R^2: 0.3380, MAPE: 420.33%). O modelo ensemble baseado na média das previsões (RMSE: 1.0494, MAE: 0.8361, R^2: 0.4879, MAPE: 377.49%) ficou entre os melhores, com destaque para o menor MAPE. Esses resultados mostram que ensembles, além de reduzirem erros, tendem a explicar melhor os dados. No entanto, métricas como RMSE e MAPE devem ser avaliadas em conjunto, já que o RMSE é mais afetado por outliers, enquanto o MAE oferece uma visão mais estável do erro médio.

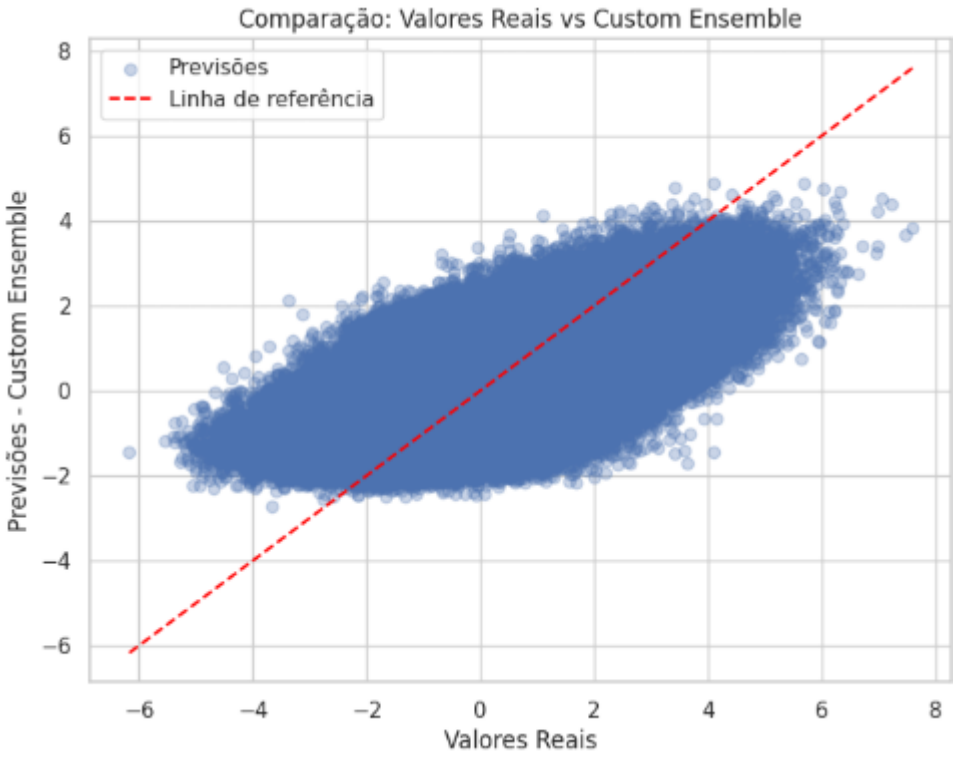
Conclusões baseadas nos resultados obtidos:

Os resultados mostram que os modelos ensemble, como Random Forest e Gradient Boosting, tiveram melhor desempenho do que os modelos individuais (Regressão Linear, SVR e KNN), com menores erros (RMSE e MAE) e maior R^2, indicando maior capacidade de explicar os dados. A análise conjunta de RMSE e MAE é importante, já que o primeiro é mais sensível a outliers, enquanto o segundo dá uma visão mais estável do erro médio. O ensemble baseado na média das previsões se destacou pelo menor MAPE, mostrando bom controle sobre os erros percentuais. Já o KNN teve o pior desempenho, reforçando que, neste caso, modelos mais sofisticados são mais adequados. No geral, os ensembles se mostraram mais eficazes em termos de precisão e robustez.


Comparativo de Desempenho entre Modelos
RMSE
1.2
1.0
0.8
0.6
0.4
0.2
0.0
LinearRegression
SVR
KNN
RandomForest
GradientBoosting
CustomEnsemble
Modelos


Curva de Aprendizado - Random Forest
RMSE
1.06
1.05
1.04
1.03
1.02
25
50
75
100
125
150
175
200
Número de Estimators


Curva de Aprendizado - Gradient Boosting
RMSE
1.12
1.10
1.08
1.06
1.04
1.02
1.00
25
50
75
100
125
150
175
200
Número de Estimators


Comparação: Valores Reais vs Custom Ensemble
Previsões
Linha de referência
Previsões - Custom Ensemble
8
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6
8
Valores Reais

Referências e links para bibliotecas/ferramentas utilizadas:

Segue abaixo a lista com as referências e os links das bibliotecas e ferramentas utilizadas no projeto:

- **Python**: https://www.python.org
- **Pandas**: https://pandas.pydata.org
- **NumPy**: https://numpy.org
- **Matplotlib**: https://matplotlib.org
- **Seaborn**: https://seaborn.pydata.org
- **Scikit-learn**: https://scikit-learn.org
- **Jupyter Notebook**: https://jupyter.org