Relatório Final — Predição de DAU case Rankmyapp

Objetivo:

Desenvolver um **modelo de Machine Learning** para realizar a predição de **Daily Active Users (DAU)** dos aplicativos

---

1. Coleta de Dados:

Os dados foram extraídos diretamente de um banco **MySQL**, com auxílio da biblioteca `pymysql`:

- **Tabelas utilizadas:**
  - `daumau` (target: `dauReal`)
  - `desinstalacoes`
  - `installs`
  - `ratings_reviews`

tratamento:

1. **Conversão de datas:** foi feita a conversão para `datetime64[ns]`.
2. **Remoção de datas futuras:** como a data da coleta no case é **11/06/2025**, foram eliminadas datas posteriores.
3. **Exclusão de colunas:**
   - `mauReal` foi removida por ser colinear com a target.
   - `country` e `lang` também foram eliminadas por falta de variabilidade.
4. **Normalização de colunas:** nomes como `appid` foram padronizados para `appId`.
5. **Tratamento de nulos:**

- Somente a variável `dauReal` apresentou nulos, e como ela é a target, **linhas com valores ausentes foram removidas**.

6. **Remoção de duplicatas por `appId` e `date`.**
7. **Merge das tabelas** em uma base única chamada `merged_df`, com **join do tipo left**.

unir as tabelas com intuito de agregar informações à tabela com a target.

Shape: (40851, 10)

	appId	date	dauReal	predictionLoss	newinstalls	category	ratings	daily_ratings	reviews	daily_reviews
0	com.app.17806	2024-01-01	163431.0	27040.0	5955.0	OTHERS	682224.0	154.0	168471.0	57.0
1	com.app.80610	2024-01-01	611934.0	112183.0	10890.0	BUSINESS	893104.0	191.0	359446.0	120.0
2	com.app.62062	2024-01-01	22618.0	4554.0	871.0	SHOPPING	52665.0	1.0	1845.0	2.0
3	com.app.34216	2024-01-01	1729.0	1786.0	NaN	NaN	NaN	NaN	NaN	NaN
4	com.app.26790	2024-01-01	6245.0	1758.0	679.0	FINANCE	50415.0	0.0	26728.0	1.0

8. Adicionou aos nulos o valor `0` em númericas e `DESCONHECIDO` na categórica
9. E então salvou a base como `data.csv` na pasta no drive `/content/drive/MyDrive/Codes/Rankmyapp/data.csv` a base se encontra neste repositório na pasta `data`

---

2. Análise Exploratória de Dados

A **estratégia** foi de utilizar para maior rapidez da versão 1 do modelo algoritmos que fossem mais robustos a *outliers*

A base de dados foi subdivida numa proporção 75% para o treinamento, 25% para validação.
Essa subdivisão foi feita a partir da data, uma vez que, esses dados estão ordenados por data. Portanto, a partir de 25 de julho de 2024 foi utilizado para validação.

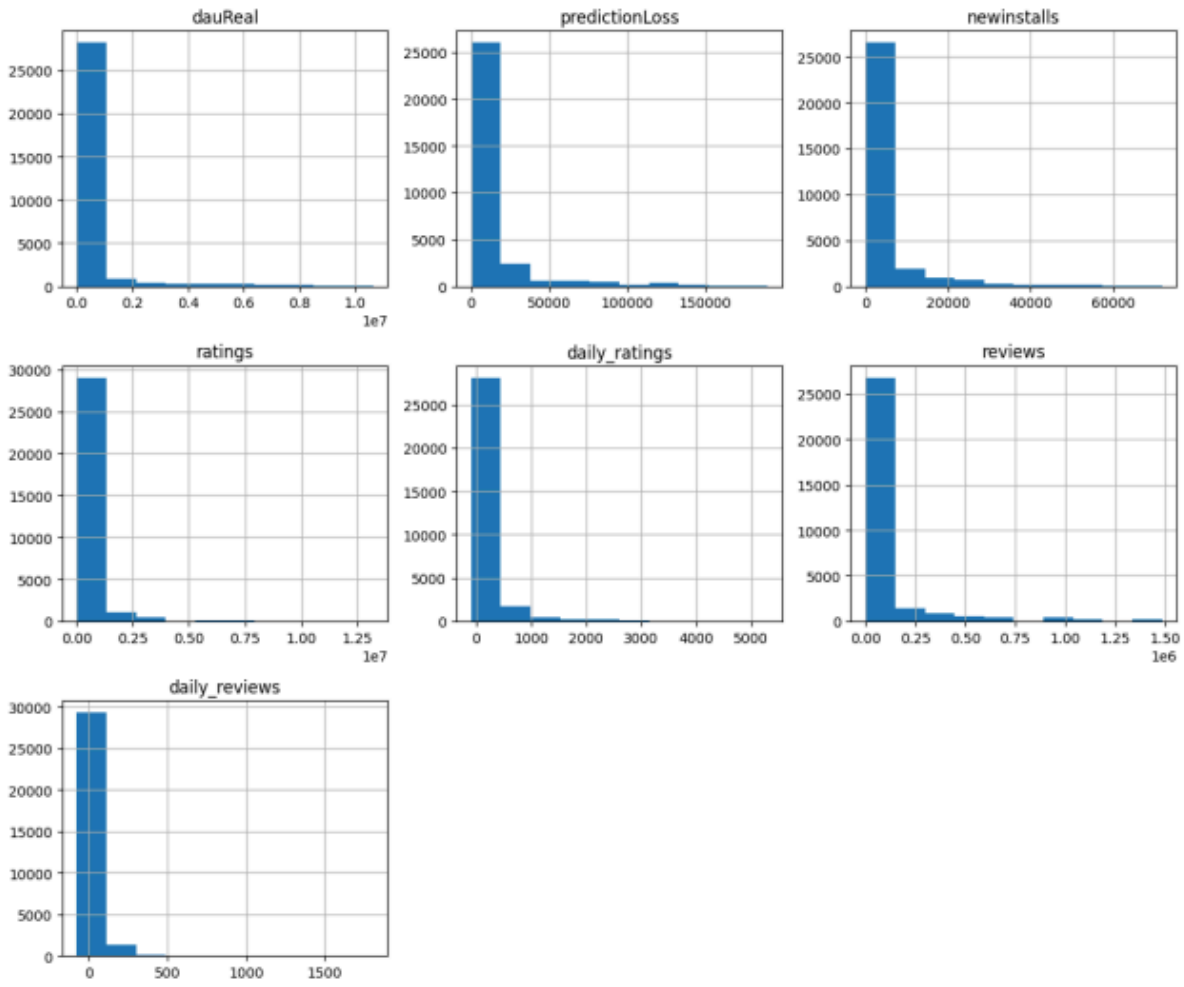
Observação: decidiu-se **não remover os outliers** inicialmente, por questões de tempo e complexidade, bem como reincidência de quais linhas são outliers, isso sugere de que esses dados se tratam de medições legitimas, em que ouveram pico de certos aplicativos . Como sugestão futura, recomenda-se:

- Teste A/B com e sem outliers.
- Aplicação de `QuantileTransformer` ou técnicas robustas de normalização.

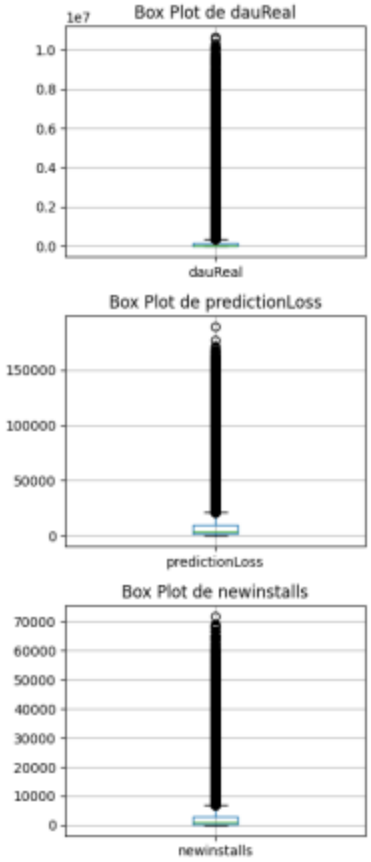
1. Foi utilizada a função baseada no método do **1.5 * IQR**:

```
def identificar_outliers_iqr(df, coluna):
    Q1 = df[coluna].quantile(0.25)
    Q3 = df[coluna].quantile(0.75)
    IQR = Q3 - Q1
    limite_inferior = Q1 - 1.5 * IQR
    limite_superior = Q3 + 1.5 * IQR
    outliers = df[(df[coluna] < limite_inferior) | (df[coluna] > limite_superior)]
    return outliers
```

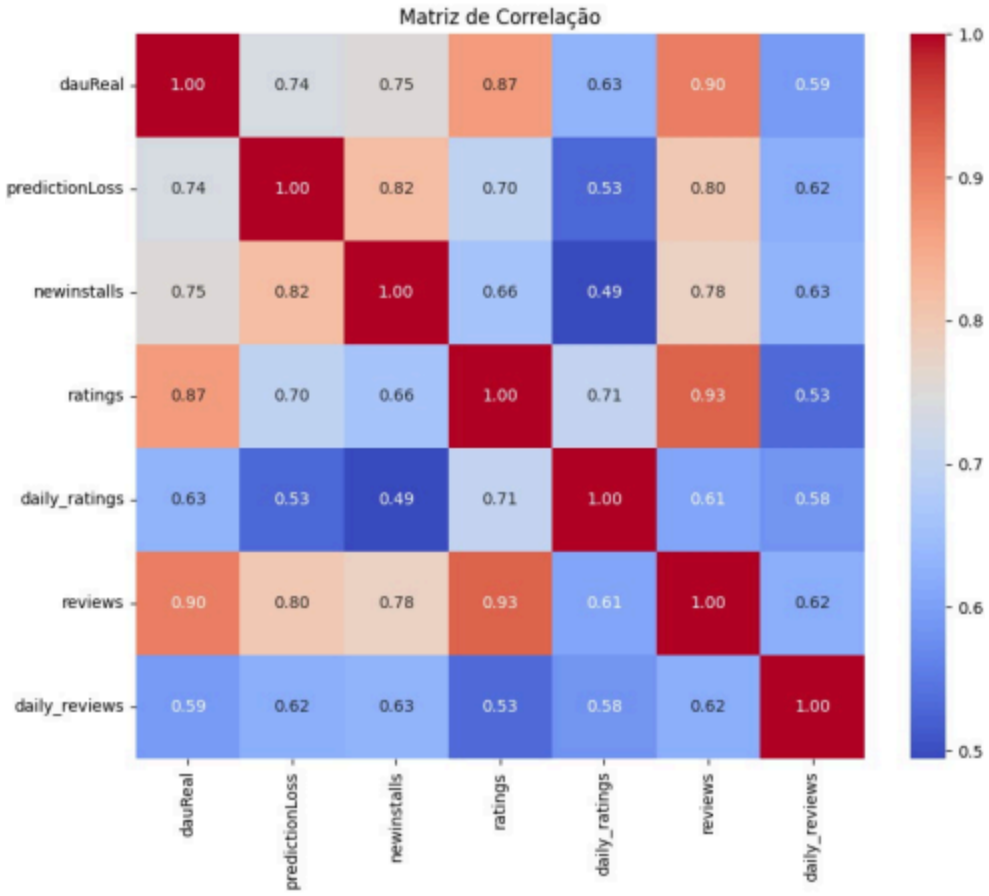
2. **Histogramas:** Foram plotados histogramas para todas as variáveis numéricas, revelando **forte assimetria à esquerda** e concentração próxima a zero.



3. **Boxplots:** evidenciaram **a presença de outliers significativos**, especialmente em `dauReal`, `predictionLoss`, `ratings`, `reviews`, entre outros.



4. **Correlação:** ao analisar a correlação podemos identificar variáveis que são altamente correlacionadas e removê-las, uma vez que agregam a mesma informação, selecionando para ficar aquela que têm maior correlação com a target.



como há alta correlação entre `reviews` e `ratings`, `predictionLoss` e `newinstalls` se manteve `reviews` e `newinstalls` e retirou-se `ratings` e

`predictionLoss`. Isso ajuda a evitar multicolinearidade em algoritmos lineares, mesmo que ensembles lidem bem com isso.

---

3. Modelagem e Treinamento

Target: `dauReal`

Procedimentos:

1. **Remoção:** Como todos os dados estão em 2024, as datsa para fatores sazonais foram descartadas.
   Retiraremos as antes discutidas e também, e os IDs para não atrapalhar o modelo
   • os ids pois pode gerar
   overfitting
   • ao aplicar
   `onehotencoding`, geraria muitas *features*
   • como há categorias, já há agrupamento para segmentar

2. **encoders:** `StandarScaler` para normalizar as numéricas e o `onehotencoder` para numerar a
   categórica

```
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numerical_features),
        ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_features)
    ],
    remainder='passthrough'
)
```

3. **Treino e teste:** 80/20 de proporção

> X_train: (24627, 11)
> X_test: (6157, 11)
> y_train: (24627,)
> y_test: (6157,)

4. **Algoritmos:** Foram utilizados 3 algoritmos de reegressão e avaliados com **RMSE** para seleção e **validação cruzada** para melhor avaliação:

- `LinearRegression`

> *utilizaremos regressão linear como um modelo de base ao comparar outros algoritmos mais*
> *robustos e menos sensíveis a outliers*

> Média da RMSE: 536157.612374299

- `RandomForestRegressor`

> *como ele é menos sensível a outlier, iremos utilizá-lo*

> Média de RMSE: 182980.8655450555

- `GradientBoostingRegressor`

> *Um 3º algoritmo sendo ensemble, o Gradient Boosting corrige os erros das árvores anteriores,*
> *sendo menos propenso a overfitting*

> Média de RMSE: 233304.58764601886

Observou-se que o **Random Forest** apresentou melhor desempenho e por isso foi selecionado.

5. **Tunamento:** Não realizado devido à complexidade dentro do tempo. Como sugestão ficou a utilização de `GridSearchCV`:

```
from sklearn.model_selection import GridSearchCV
param_grid = {
 'n_estimators': [100, 200],
```

```
 'max_depth': [None, 10, 20],
 'min_samples_split': [2, 5],
 'min_samples_leaf': [1, 2],
 'bootstrap': [True, False]
}
grid_search = GridSearchCV(RandomForestRegressor(random_state=42),
param_grid=param_grid, cv=5)
grid_search.fit(X_train, y_train)
print("Melhores parâmetros:", grid_search.best_params_)
print("Melhores Estimadores:", grid_search.best_estimator_)
```

- `n_estimators:` número de árvores, quanto mais, melhor performance e mais custo
- `max_depth:` para evitar overfitting e underfitting
- `min_sample_split:` mínimo de nodes que irá se subdividir
- `min_sample_leaf:` limite mínimo de folhas nas árvores (complexidade)
- `bootstrap:` Usa ou não amostragem com reposição

fonte: Geeks for Geeks

6. **Salvamento:** por fim, o modelo foi salvo utilizando a biblioteca `pickle` e pode ser encontrado na pasta `models` do repositório como `modelo_previsao_dau.pkl` .

---

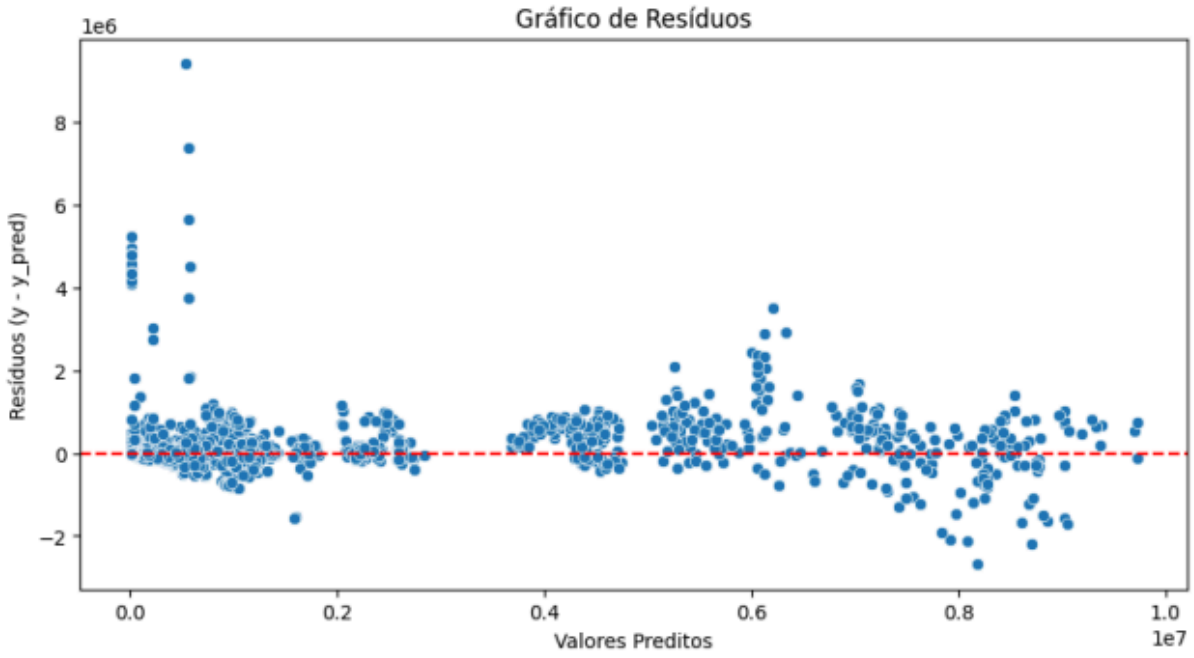
4. Validação

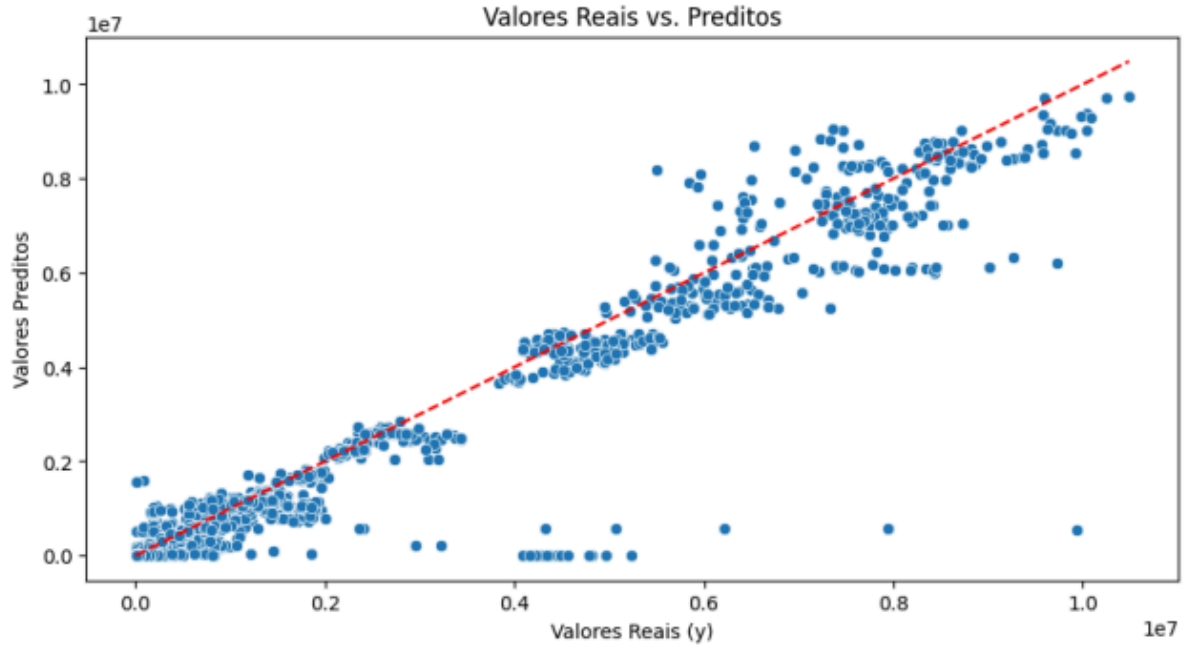
Para validar e avaliar os modelos, foi utilizado uma nova conexão com o banco de dados MySQL e se repetiu o procedimento da coleta dedados, porém desta vez se fez a subdivisão **a partir do dia 25 de julho de 2024** para que dados que não fizeram parte do treinamento fossem utilizados. Os moesmos foram processados na preparação de maneira semelhante à modelagem feita anteriormente.

📈 Métricas Obtidas

- **R^2:** 0.95
- **RMSE:** 309.492
- **MAE:** 78.508
- `R2:` coeficiente de determinação. Percentual da variância dos dados que é explicado pelo
  modelo
- `MAE:` média da diferença entre o valor real com o predito
- `RMSE:` penalisa as grandes diferenças, quanto muito distante de MAE, indica que outliers
  estão afetando o modelo.

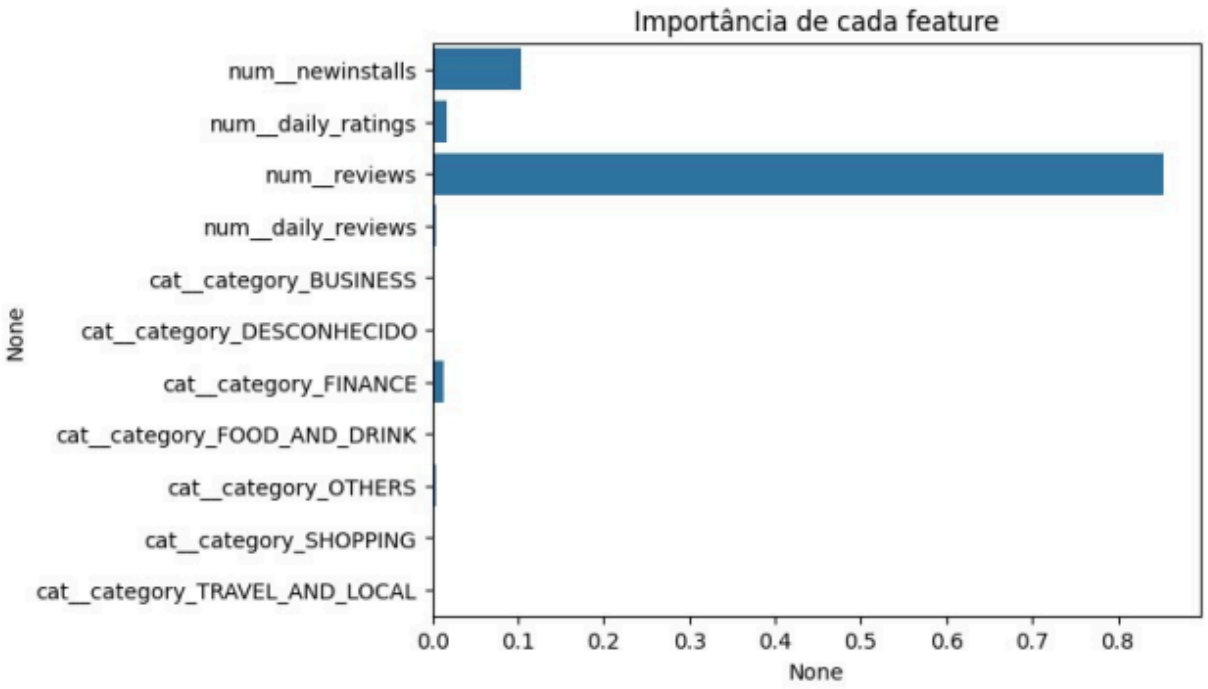
avaliação gráfica dos resíduos:





R-squared com 95% indica bom desempenho. RMSE ser muito maior que MAE conclui-se que
há alguns erros maiores (outliers). Os erros grandes são provavelmente pela existência de
muitos zeros e outliers. A análise gráfica demostra que o modelo segue a maioria da disperção
dos dados

Features mais influentes:



Observa-se que o número de reviews determina cerca de 85% DAU neste modelo, e o número
de instalações é o 2º mais importante.

---

5. Conclusão

sobre perfomance:

Concluimos que modelo precisa passar por melhorias, apesar de apresentar bom desempenho
pela métrica do
`R2`, os outliers e presença de valores muitos baixos ou zero leva a maiores erros, o que é reportado pela métrica
`RMSE`. Pela a análise gráfica, observa-se que o modelo obteve
bom ajuste, mas se reforça o melhor tratamento de outliers e de
testes A/B para que se
consiga um desempenho com menores erros.

sobre melhorias:

Devido ao tempo do case, observei melhorias que precisaria de mais para testá-las. Aqui estão
melhorias que devem ser testadas para construção de um modelo melhor:

Outliers: aplicar a remoção ou tratamento de outliers como o `quantile_transformer`, aplicando testes A/B

Seleção de features: utillizar metodologias de seleção e rankeamento de features como `Boruta`,
e
`RFE`. Úteis para algoritmos *ensembles*.

segmentação dos ids dos apps: uma melhor segmentação dos apps para agregar informação no
treinamento sem aumentar muito a dimensionalidade.

Normalização dos dados: Também pode haver teste com o `Robust Scaler` para a normalização na preparação dos dados, uma vez que ele lida melhor com outliers

Remoção de nulos: remover linha com muitos valores nulos da base montada na coleta dos
dados, e realizar testes A/B de performance

Tunamento: aplicação de ajustes nos hiperparâmetros do modelo

Algoritmos: Testar outros algoritmos, como `XGBoost`