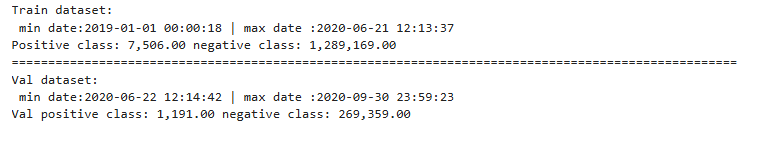
**Projeto de Detecção de Fraude**

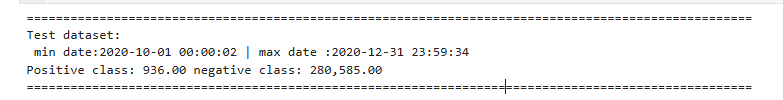
O objetivo deste projeto foi treinar um modelo de Machine Learning para identificar transações fraudulentas e classificá-las corretamente, ao mesmo tempo em que minimiza o erro de classificar transações legítimas como fraudulentas. O dataset utilizado foi extraído do Kaggle, disponível [aqui](https://www.kaggle.com/datasets/kartik2112/fraud-detection).

**Divisão dos Dados: Treinamento, Validação e Teste**

Considerando que este é um problema de séries temporais, os dados foram divididos da seguinte forma:

* **Treinamento**: De 01/01/2019 a 21/06/2020, com 7.506 transações fraudulentas e 1.289.169 transações legítimas.
* **Validação**: De 22/06/2020 a 30/09/2020, com 1.191 transações fraudulentas e 269.359 transações legítimas.
* **Teste**: De 01/10/2020 a 31/12/2020, com 936 transações fraudulentas e 280.585 transações legítimas.



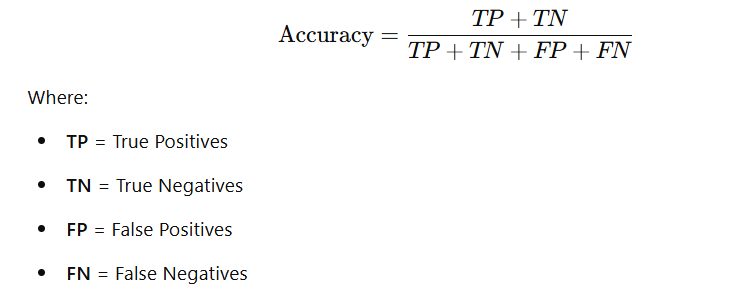


O objetivo é identificar a maior quantidade possível de fraudes e classificar corretamente as transações legítimas, mantendo o equilíbrio entre as classes.

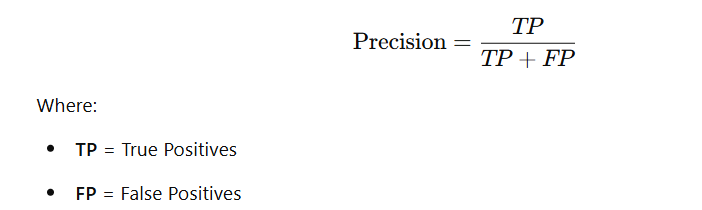
**Métrica de avaliação**

As métricas utilizadas para avaliar o modelo são:

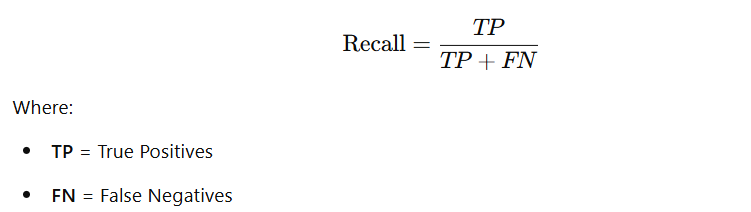
**Acurácia**: Mede a porcentagem de previsões corretas entre todas as previsões feitas. Em um cenário de fraude, onde a maioria dos exemplos são de transações "não fraudulentas", uma alta acurácia pode ser alcançada simplesmente prevendo "não fraude" o tempo todo, o que não seria útil.



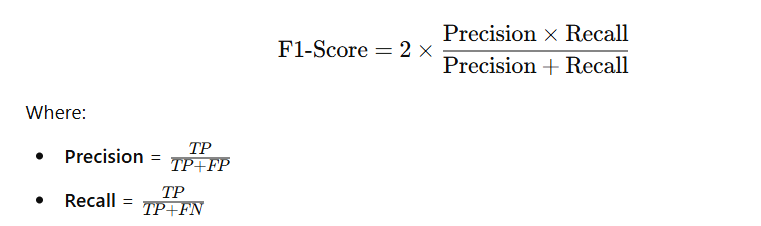
**Precision:** Mede a proporção de transações que foram classificadas como fraude e realmente são fraude.



**Recall:** Mede a proporção de fraudes que foram corretamente identificadas pelo modelo. Ou seja, de todas as transações fraudulentas, quantas o modelo conseguiu identificar como fraude.



**F1-Score**: É a média harmônica entre **precisão** e **recall**, sendo uma boa métrica para equilibrar ambos.



**AUC (Área sob a Curva ROC)**: Mede a capacidade do modelo de distinguir entre as classes (fraude vs. não fraude) em todos os limiares possíveis. A curva ROC (Receiver Operating Characteristic) plota a taxa de verdadeiros positivos (recall) versus a taxa de falsos positivos (1 - precisão). A **AUC** varia de 0 a 1, onde 1 indica uma perfeita distinção entre as classes.

A métrica principal escolhida foi o **F1-Score**, pois ele proporciona um bom equilíbrio entre a detecção de fraudes e a minimização de falsas classificações de transações legítimas como fraudulentas.

**Modelos escolhidos**

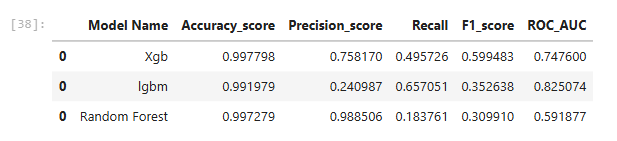
Foram testados os seguintes modelos:

* **Random Forest**
* **XGBoost**
* **LightGBM (LGBM)**

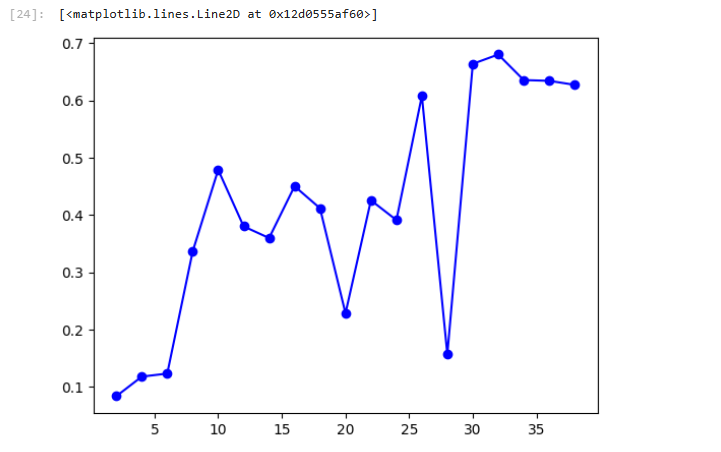
**Metodologia**

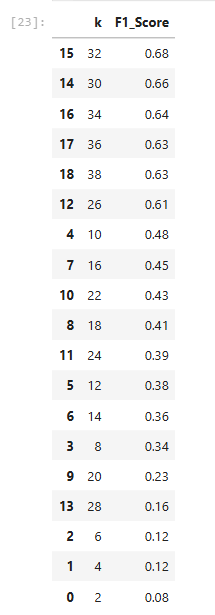
**Baseline**

O primeiro passo foi treinar os modelos com os dados originais, sem nenhuma feature adicional, e utilizar os dados minimamente processados. Esse treinamento inicial gerou o **baseline**. O modelo com o melhor F1-Score foi o **XGBoost**, mas devido ao tempo de treinamento longo ao ajustar os hiperparâmetros e selecionar features, optei por usar o **LGBM**, que apresentou um bom desempenho, mas é mais leve e eficiente para grandes volumes de dados.

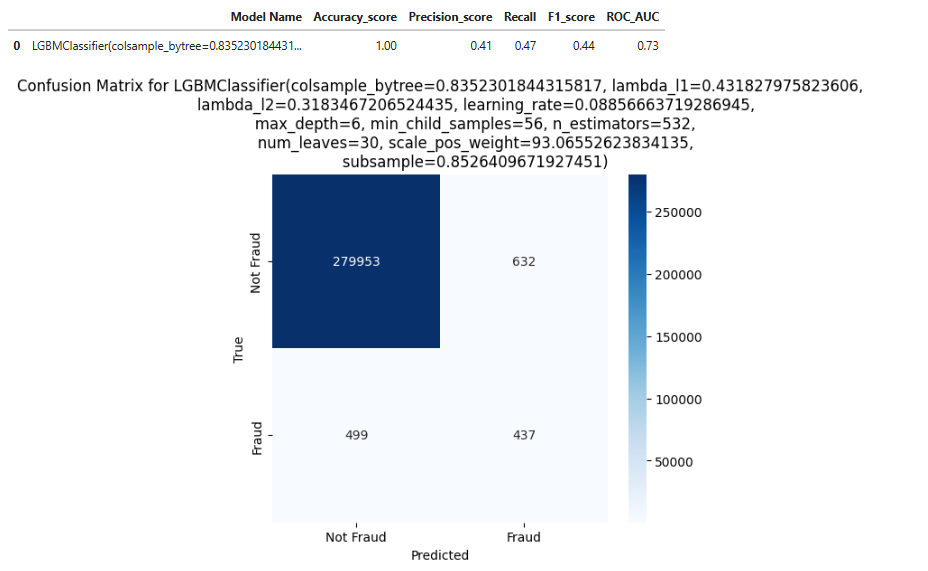


**Versão 1**

Na primeira versão, criei features temporais, como a diferença de tempo entre transações e o número de transações diárias. Utilizei o **SelectKBest** do scikit-learn para selecionar as features mais relevantes após o ajuste dos hiperparâmetros usando o **Optuna**. Embora parecesse contraintuitivo tunar os hiperparâmetros antes de selecionar as features, pesquisas indicam que a combinação de ambos os processos pode gerar um modelo mais eficiente, pois o modelo ajustado pode descartar ou incluir features de forma mais precisa. Após testar várias combinações de **k**, a melhor seleção foi de **32 features**.

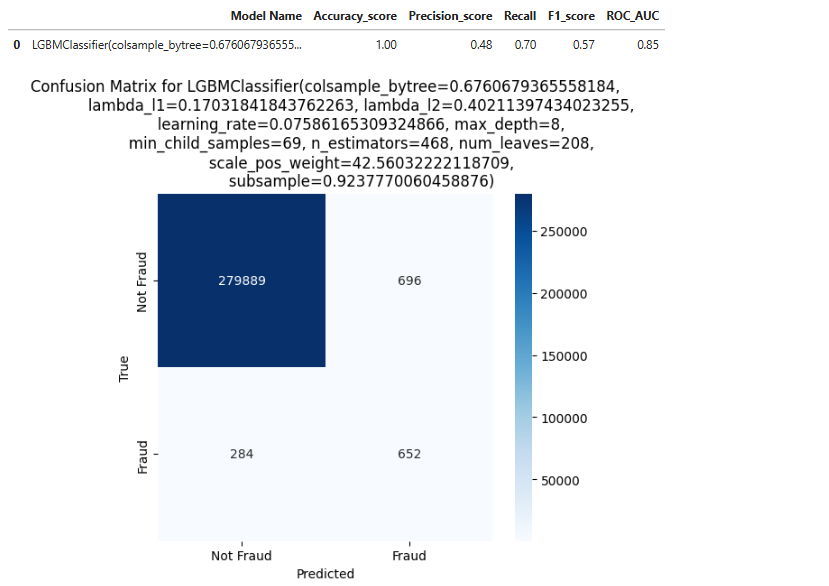


Na primeira versão consegui um f1-score de **0.44** após tunar os hiperparametros.



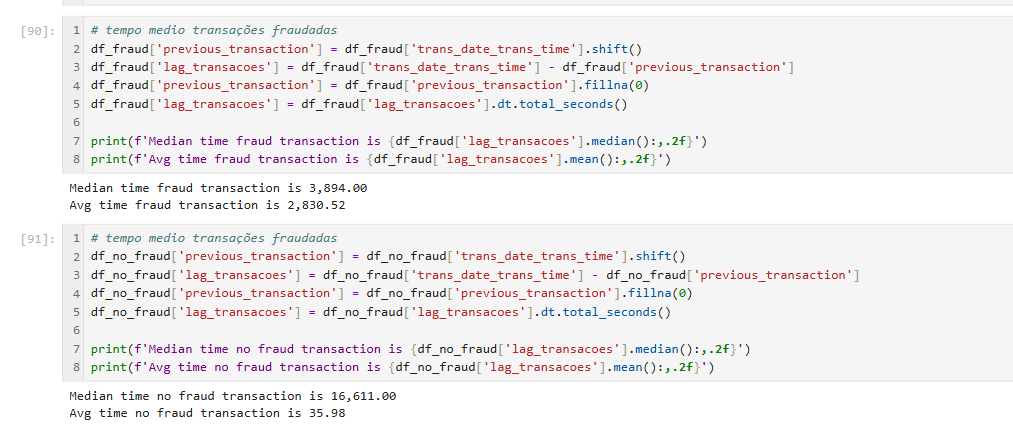
**Versão 2**

Na versão 2 o que fiz foi derivar algumas features, tunar o modelo e por fim selecionar as features mais relevantes usando o selectkbest do sklearn. Neste versão obtive um score de **0.57** nos dados de teste.



**Versão 3**

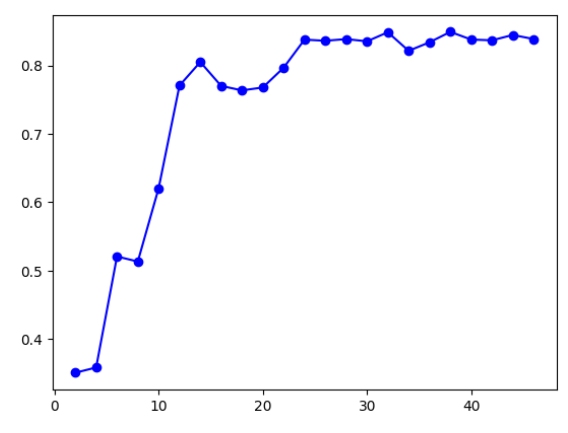
Antes de passar para a terceira versão, analisei o comportamento das transações fraudulentas e percebi que as fraudes tendem a ocorrer com maior intervalo de tempo entre as transações, além de terem um valor médio maior que as transações legítimas. Com base nisso, derivei novas features, como o tempo médio entre transações e a média/desvio padrão do valor das transações.

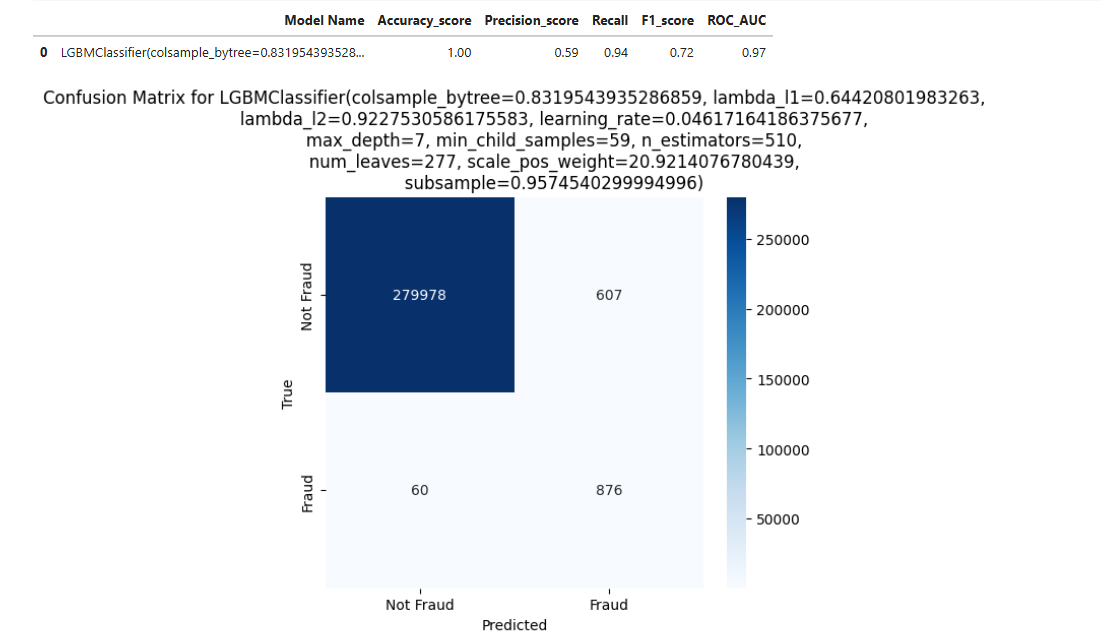


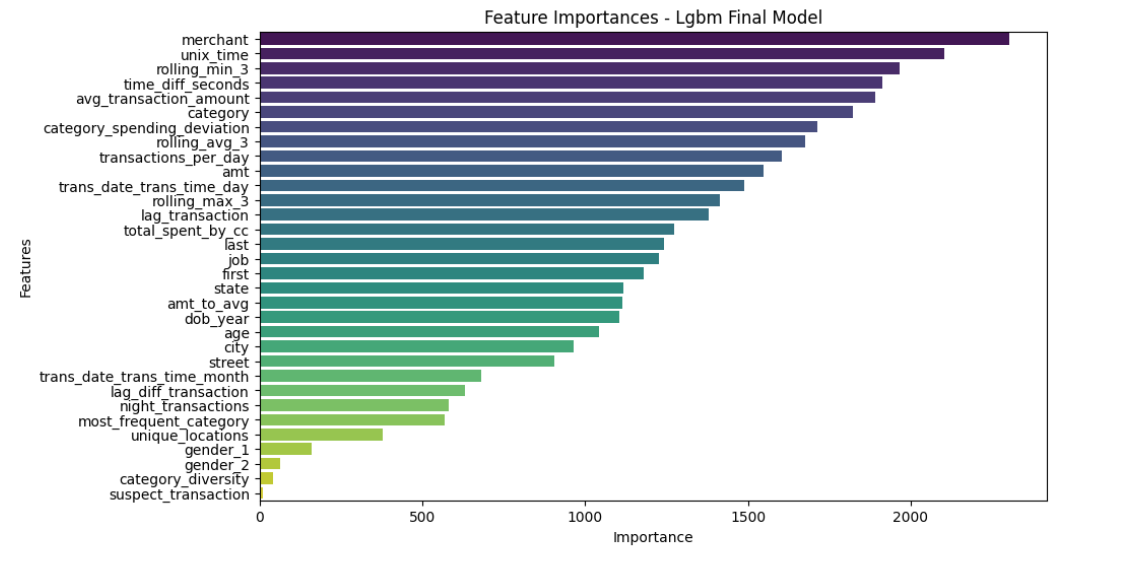


Após ajustar os hiperparâmetros novamente e realizar uma nova seleção de features, obtive um **F1-Score de 0,72** nos dados de treino. A criação das novas features foi crucial para esse aumento de desempenho.

Uma das ideias que utilizei para a seleção das features veio de um vídeo do Kaggle Grandmaster **Mario Filho**, que sugere que o *feature engineering* deve considerar agregações, médias móveis e lags entre datas. Link do vídeo: <https://www.youtube.com/watch?v=ft77eXtn30Q&ab_channel=MarioFilhoEnglish>.







**Conclusão e próximo passos**

Este projeto destacou a importância de criar novas features para melhorar o desempenho do modelo. O aumento no **F1-Score** evidenciou o impacto positivo do **feature engineering**. Contudo, ainda há espaço para melhorias, especialmente na seleção de features. Em ciclos futuros, pretendo explorar ferramentas como **Featurewiz** e **Boruta** para realizar a seleção de features de forma mais eficaz, comparando os resultados e testando qual abordagem gera o melhor desempenho.