Atividade 1 Aprendizado Supervisionado

Relatório de Atividade

Julio Vinicius Amaral Oliveira
Matrícula: 230537
Disciplina: MO432
Professor: Marcos M. Raimundo

10 de maio de 2025

Sumário

1 Introdução

- O objetivo do trabalho é aplicar técnicas básicas de aprendizado supervisionado para detectar fraudes em transações financeiras. O trabalho foi dividido em duas partes: a primeira parte vamos implementar dois modelos de detecção de fraude, enquanto a segunda parte resolveremos algumas questões teóricas relacionadas ao aprendizado supervisionado.
- O dataset utilizado é o Credit Card Fraud Detection. Ele possui transações realizadas com cartões de crédito em setembro de 2013. O conjunto de dados tem a maioria de suas features anonimizadas, com exceção de algumas variáveis como: tempo, valor da transação e a variável que indica se a transação é fraude ou não. Uma característica bem importante desse datasete é que ele é muito desbalanceado, com apenas 0.172% das transações sendo fraudes.

2 Metodologia (Parte Prática)

2.1 Análise Exploratório de Dados (EDA) e Pré-processamento

Como dito anteriormente, o dataset é bastante desbalanceado, o primeiro passo realizado foi verificar a quantidade de fraudes e não fraudes utilizando o método `value_counts()` do pandas. Obtendo o seguinte resultado:

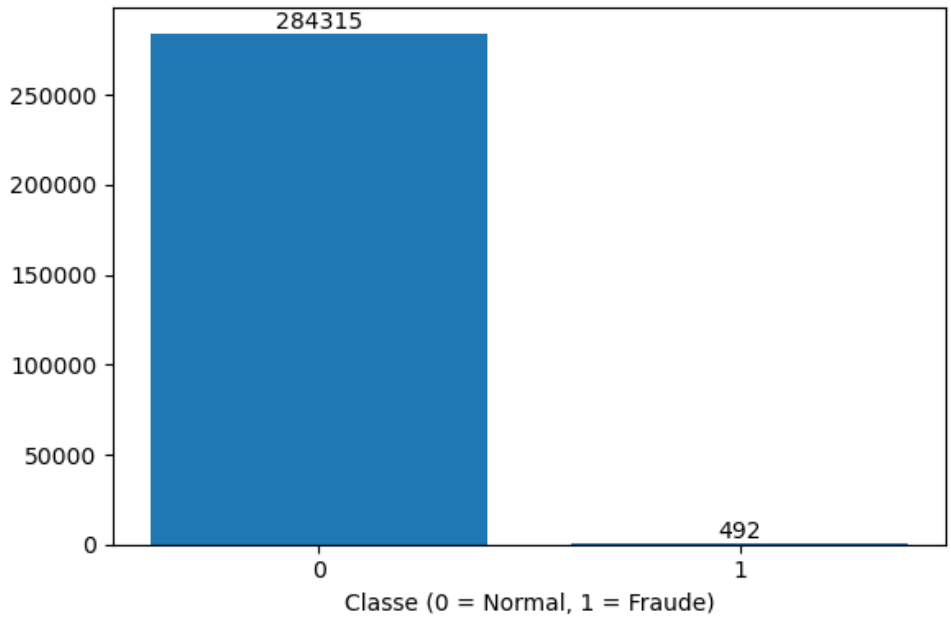


Figura 1: Proporção de fraudes e não fraudes no dataset

Após isso, procuramos entender como as features se relacionavam com a coluna de Class, que era nossa coluna alvo. Então usamos o método `corr()` do pandas, buscando a correlação entre Class e o restante das features. obtendo o seguinte resultado para as features com maior correlação absoluta:

Feature	Correlação com Class
V17	-0.326481
V14	-0.302544
V12	-0.260593
V10	-0.216883
V16	-0.196539
V3	-0.192961
V7	-0.187257

Tabela 1: Correlação das features com Class

2.2 Abordagem 1: Modelagem da Classe Normal (Detecção de Anomalia)

Para a primeira abordagem utilizamos o modelo de detecção de anomalias. Primeiramente, separamos apenas as 3 features mais relevantes para o modelo, esse número foi escolhido com base na performance do modelo. Foi percebido que ao diminuir cada vez mais o número de features tanto a medida de AUC-PR quanto a matriz de confusão resultavam em valores melhores. Após isso, separamos 2000 transações normais, e com o StandartScaler aplicamos uma normalização para esses dados. Depois aplicamos a normalização para o conjunto inteiro de dados (Com exceção da coluna Class). Após isso, tentamos encontrar o melhor valor de bandwith usando o método `GridSearchCV` do sklearn. O melhor valor encontrado foi 0.479, e com isso aplicamos um KDE e calculamos o score de anomalias.

	anomaly_score	true_class
count	284315.000000	284315.0
mean	3.818309	0.0
std	4.005647	0.0
min	2.346846	0.0
25%	2.822756	0.0
50%	3.311860	0.0
75%	4.210312	0.0
max	557.518645	0.0

Tabela 2: Transações normais

	anomaly_score	true_class
count	492.000000	492.0
mean	162.766740	1.0
std	250.679879	0.0
min	2.780611	1.0
25%	26.711496	1.0
50%	68.714390	1.0
75%	156.108730	1.0
max	1377.747682	1.0

Tabela 3: Transações fraudulentas

Apos isso, tentamos encontrar o melhor limiar para separar as transações, utilizamos o método precision_recall_curve do sklearn, e encontramos o melhor limiar para maximizar f1 sendo 34.9 Com isso, conseguimos os seguintes resultados após plotar a matriz de confusão:

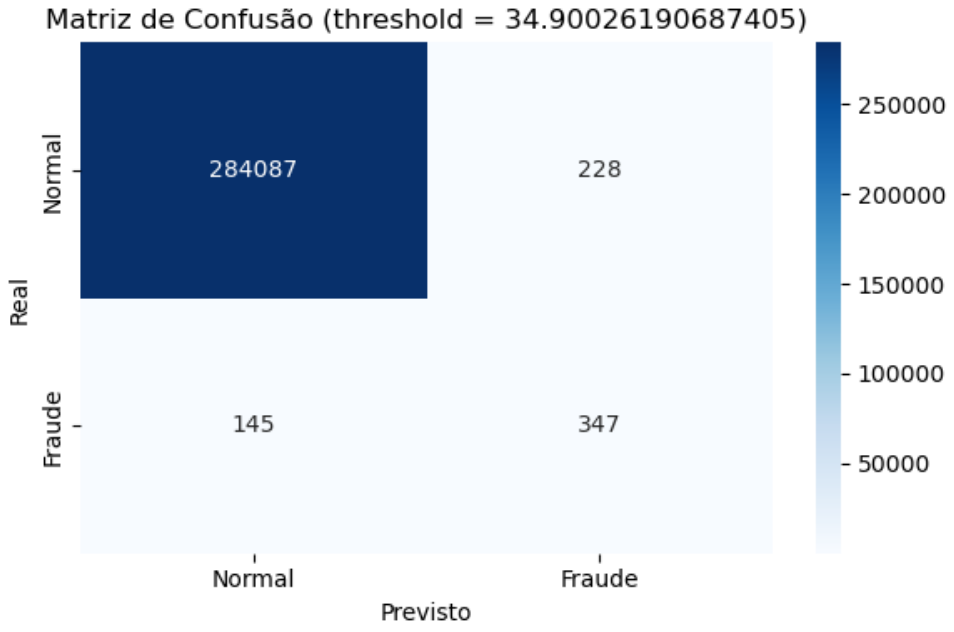


Figura 2: Matriz de confusão

Segundo o próprio repositório do Kaggle, a matriz de confusão não é o melhor indicador de desempenho quando temos classes desbalanceadas e nesses casos é recomendado o uso da área abaixo da curva de precision-recall. Cujo resultado foi de 0.623

2.3 Abordagem 2: Classificação Supervisionada

Nessa segunda abordagem utilizamos um modelo de regressão logística, utilizamos o método train_test_split do sklearn para separar 80% dos dados para treino e 20% para teste, isso foi realizado nos dados já normalizado com a ajuda do StandartScaler. Após isso, aplicamos o modelo de regressão logística e aplicamos o modelo nos dados de testes, obtendo a seguinte AUC-PR: 0.742, que foi um valor muito maior do que o obtido na primeira abordagem. Para ilustrar também o resultado, plotamos a matriz de confusão:

	Classe 0	**Classe 1**	**Acurácia**	**Média Macro**	**Média Ponderada**
Precision	0.999	0.829	0.999	0.914	0.999
Recall	1.0	0.643	0.999	0.821	0.999
F1-score	1.0	0.724	0.999	0.862	0.999
Support	56864	98	0.999	56962	56962

Tabela 4: Métricas de avaliação do modelo para as classes 0 (normal) e 1 (fraude)

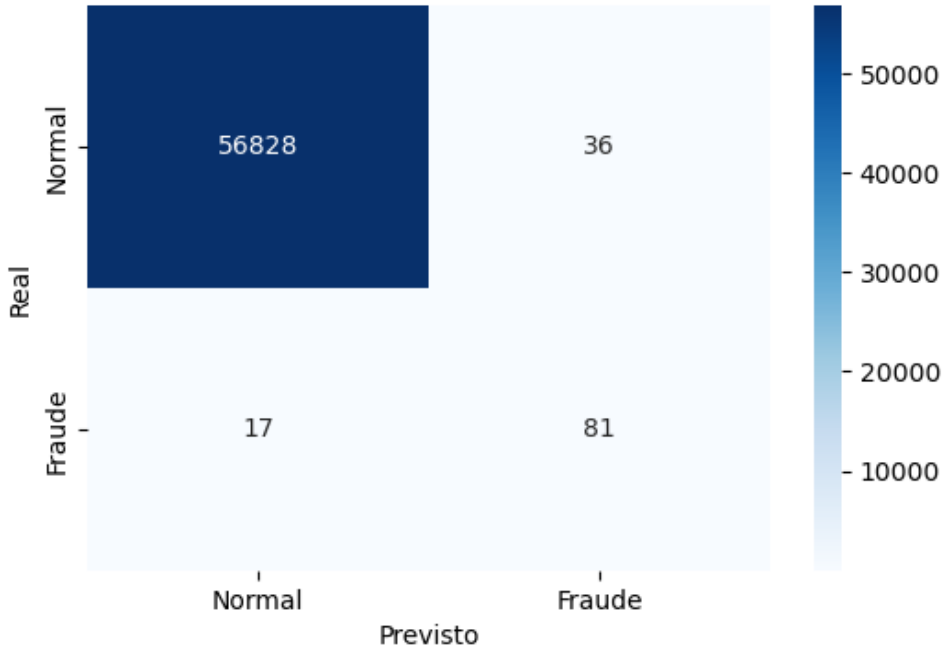


Figura 3: Matriz de confusão regressão logística

Para melhorar ainda mais o resultado, resolvemos encontrar um threshold utilizando o método `precision_recall_curve` do sklearn, e ao fazer isso encontramos um valor melhor de recall, entretanto o valor da precision diminuiu um pouco. Mas dado o contexto que o modelo deveria ser usado, ter um recall alto é melhor pois evita perdas financeiras para a empresa que utiliza o modelo, o valores obtidos foram

	Classe 0	**Classe 1**	**Acurácia**	**Média Macro**	**Média Ponderada**
Precision	0.999	0.692	0.999	0.846	0.999
Recall	0.999	0.827	0.999	0.913	0.999
F1-score	1.000	0.753	0.999	0.877	0.999
Support	56864	98	56962	56962	56962

Tabela 5: Métricas de avaliação do modelo (valores aproximados para 3 casas decimais)

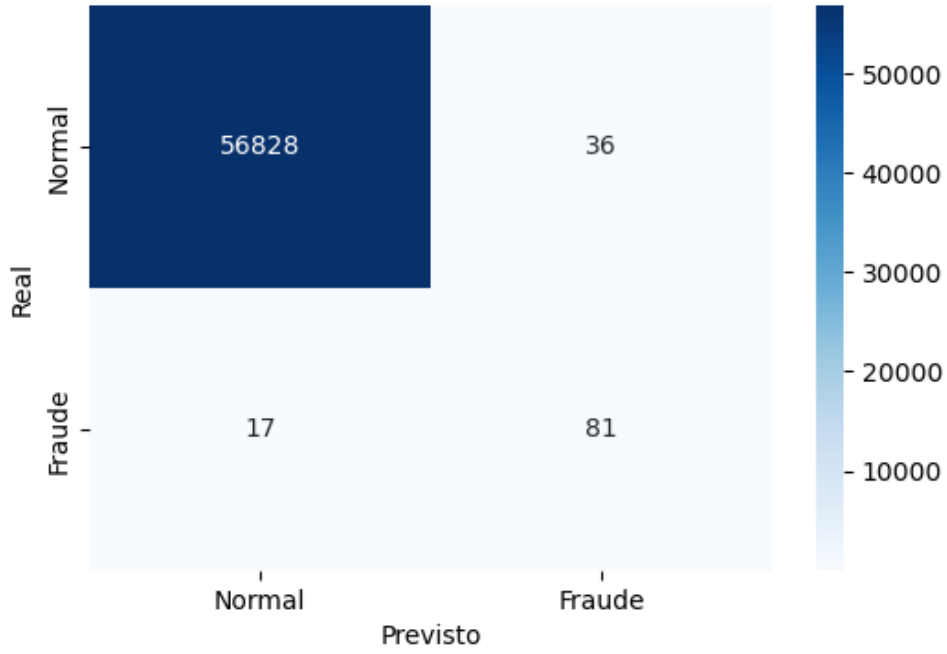


Figura 4: Matriz de confusão regressão logística

2.4 Uso de LGBM

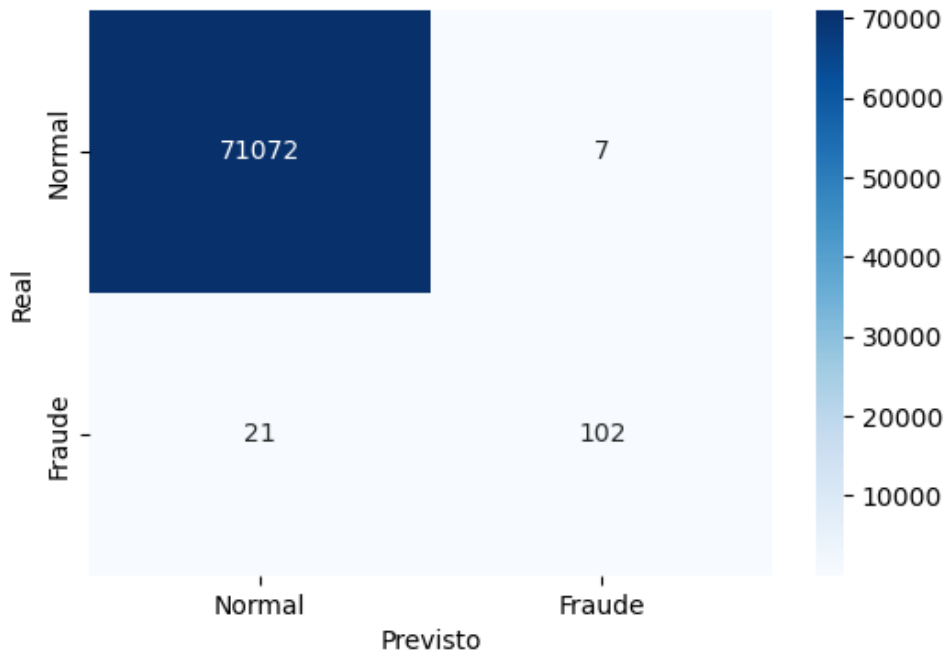
Ao utilizar o modelo LGBM, obtivemos um resultado muito superior aos anteriores. Calculamos os scores utilizando o `predict_proba` e ajustamos o threshold para maximizar o F1-score. Os thresholds, precision e recall foram encontrados utilizando o método `precision_recall_curve` e calculamos o F1 através da fórmula:

$$F1 = \frac{2 \cdot (precision \cdot recall)}{precision + recall} \tag{1}$$

Seguem os resultados de precisão, recall e F1-score:

	0	**1**	**Accuracy**	**macro avg**	**weighted avg**
Precision	1.000	0.936	1.000	0.968	1.000
Recall	1.000	0.829	1.000	0.915	1.000
F1-score	1.000	0.879	1.000	0.940	1.000
Support	71079	123	1.000	71202	71202

Tabela 6: Métricas de avaliação do modelo LGBM



Também calculamos a AUC-PR, que foi de 0.8585 A utilização do LGBM trouxe uma distribuição muito heterogênea entre os scores, oque pode ser visto na figura 5. Enquanto os scores de dados normais se concentram em torno de 0, os scores de dados fraudulentos se concentram em torno de 1, mostrando que o LGBM conseguiu separar bem as classes.

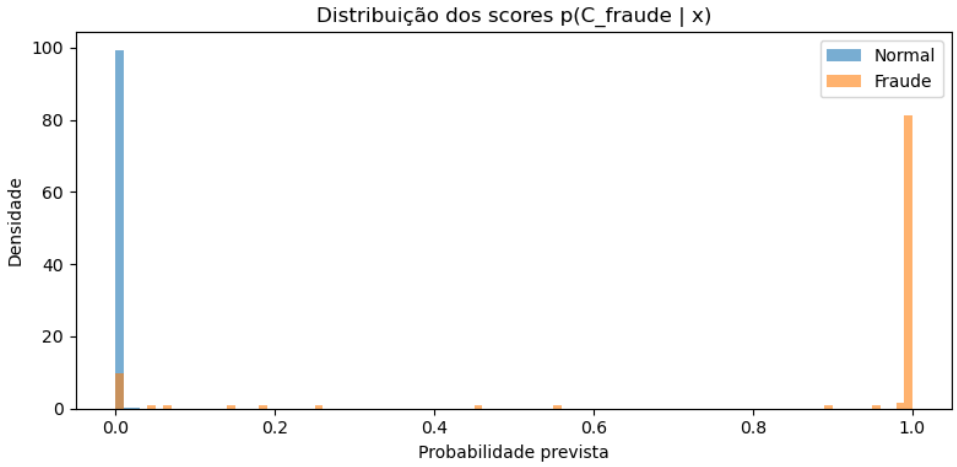


Figura 5: Distribuição dos scores do modelo LGBM

3 Conclusão

A primeira abordagem, que utilizou o modelo de detecção de anomalias, obteve um resultado de AUC-PR de 0.623, enquanto a segunda abordagem, que utilizou o modelo de regressão logística, obteve um resultado de AUC-PR de 0.742. Por fim, a terceira abordagem, que utilizou o modelo LGBM, obteve um resultado de AUC-PR de 0.8585. Isso mostra que o modelo LGBM foi o mais eficaz para detectar fraudes no dataset. Deve-se levar em conta também, que na primeira abordagem, o resultado só foi tão alto, devido ao fato de que reduzimos consideravelmente a quantidade de features, pois em testes prévios,

ao utilizar todas as features, os resultados eram muito ruins, com AUC-PR próximos de 0.1. A abordagem 2 pareceu melhor, tanto em questão de resultado quanto em questão de performance, já que para realizar a abordagem 1, tivemos que separar um conjunto inferior de dados, utilizar menos features para chegar em um resultado minimamente aceitável. Já na abordagem 2, utilizamos todas as features e mesmo assim chegamos em um resultado muito bom. Quanto a escolha do limiar, utilizamos o método `precision_recall_curve` para encontrar o F1 com maior valor, em um cenário real, talvez o ideal fosse utilizar um limiar que favorecesse ainda mais o recall, dado que estamos lidando com fraudes que podem significar perda financeira para uma empresa, mas ao mesmo tempo é importante considerar que um modelo que tem uma precisão baixa pode causar problemas tanto para clientes que possam ter transações bloqueadas, causando insatisfação quanto para um time que precisará investigar essas transações.

4 Respostas Teórico-Conceituais (Parte 2)

4.1 Exercício 1: Fronteira de decisão em modelos generativos

4.2 Exercício 2: Verossimilhança, entropia cruzada e regressão logística

4.3 Exercício 3: Decomposição viés-variância

4.4 Exercício 4: Avaliação em dados desbalanceados

5 Referências

A Correlação das Features com Class

Feature	Correlação com Class
V17	-0.326481
V14	-0.302544
V12	-0.260593
V10	-0.216883
V16	-0.196539
V3	-0.192961
V7	-0.187257
V11	0.154876
V4	0.133447
V18	-0.111485
V1	-0.101347
V9	-0.097733
V5	-0.094974
V2	0.091289
V6	-0.043643
V21	0.040413
V19	0.034783
V20	0.020090
V8	0.019875
V27	0.017580
Time	-0.012323
V28	0.009536
V24	-0.007221
Amount	0.005632
V13	-0.004570
V26	0.004455
V15	-0.004223
V25	0.003308
V23	-0.002685
V22	0.000805

Tabela 7: Correlação das features com a variável Class