REDES NEURAIS E DEEP LEARNING

Uso de CNNs para detecção de perfis suspeitos de fraude

Cassius Figueiredo

Outubro/2020

Alguns dados sobre pagamentos

- Cartões movimentaram R$1,84 trilhão em 2019 (Abecs). Compras com cartão de crédito superaram a marca de R$1 trilhão pela primeira vez e pagamentos digitais representaram 43% do consumo das famílias brasileiras no mesmo ano.
- Fraudes financeiras geraram um prejuízo de cerca de R$ 1,8 bilhão em 12 meses (ago/2019).

Tipos de perdas financeiras

Perdas por fraude

- **Problema**: minimização de perdas geradas por processos fraudulentos.
- **Foco**: companhia e clientes.
- **Abordagem**: uso de dados transacionais e cadastrais para modelagem que leve à identificação de potenciais perdas financeiras geradas por tecnologias de fraude.
- **Modelagem**: Random Forest básica como referência, Deep Learning (CNN) como experimento.
- **Desafio**: problema originalmente desbalanceado. O balanceamento foi ajustado com a abordagem sugerida.

Modelagem

- As abordagens tradicionais para atacar o problema da identificação de clientes com perfil fraudulento incluem regressão logística, SVM, Random Forest e XGBoost.
- Baseado no resultado de algumas referências, vamos fazer uma abordagem utilizando Redes Neurais Convolucionais e avaliar como performa em relação a uma Random Forest.

Referências

- **Xuetong Niu, Li Wang e Xulei Yang em “A Comparison Study of Credit Card Fraud Detection: Supervised versus Unsupervised” (2019),** discutem diversas abordagens para o problema.
- **Kang Fu, Dawei Cheng, Yi Tu e Liqing Zhang em “Credit Card Fraud Detection Using Convolutional Neural Networks” (2016)**, utilizam uma abordagem de janelas de tempo onde mapeiam as variáveis, convertendo em imagens e abordando o problema com CNNs.
- **Alae Chouiekha eEL Hassane Ibn EL Haj em “ConvNets for Fraud Detection analysis” (2018)**, alimentam uma CNN com imagens que representam o perfil do cliente, para classificação.

Abordagem

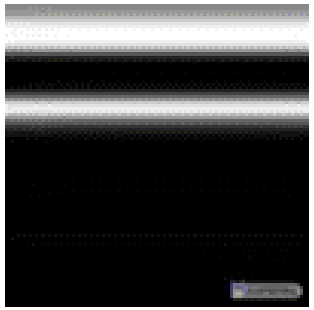
- A abordagem sugerida aqui representa um conjunto de dados fictícios de clientes com uma imagem 2D em tons de cinza, incluindo clientes com padrões saudáveis e não saudáveis (fraudulentos).
- O modelo é então alimentado com os dados etiquetados e a classificação é feita.
- Um modelo Random Forest “vanilla” foi utilizado como modelo de referência, para comparação de resultados.

Base

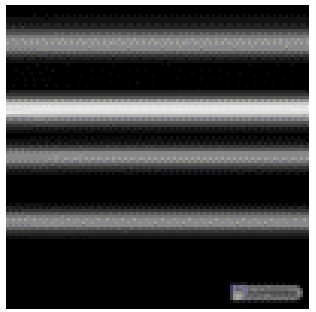
- As variáveis consistem em um conjunto de dados transacionais e cadastrais por cliente, considerado representativo para indicação de padrões fraudulentos.
- Os nomes das variáveis foram anonimizados.
- Os valores utilizados foram gerados de forma automatizada, a partir de padrões de dados próximos aos reais.
- As observações representam clientes efetivamente analisados por humanos e estão etiquetados como saudáveis ou não.

Conversão para imagens

- As 19 variáveis foram convertidas para imagens 2D de tamanho 38x38.
- Cada uma das variáveis foi normalizada para ser representada com tons de cinza e “esticada”, formando uma linha de um quadrado.



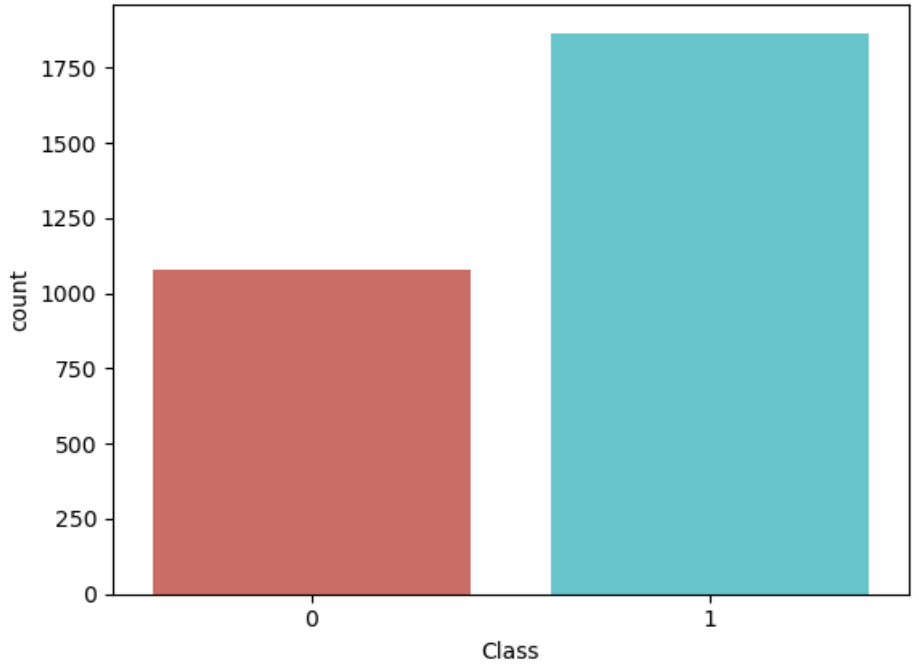
Sequência de 20 imagens representando clientes fraudulentos



Sequência de 20 imagens representando clientes saudáveis

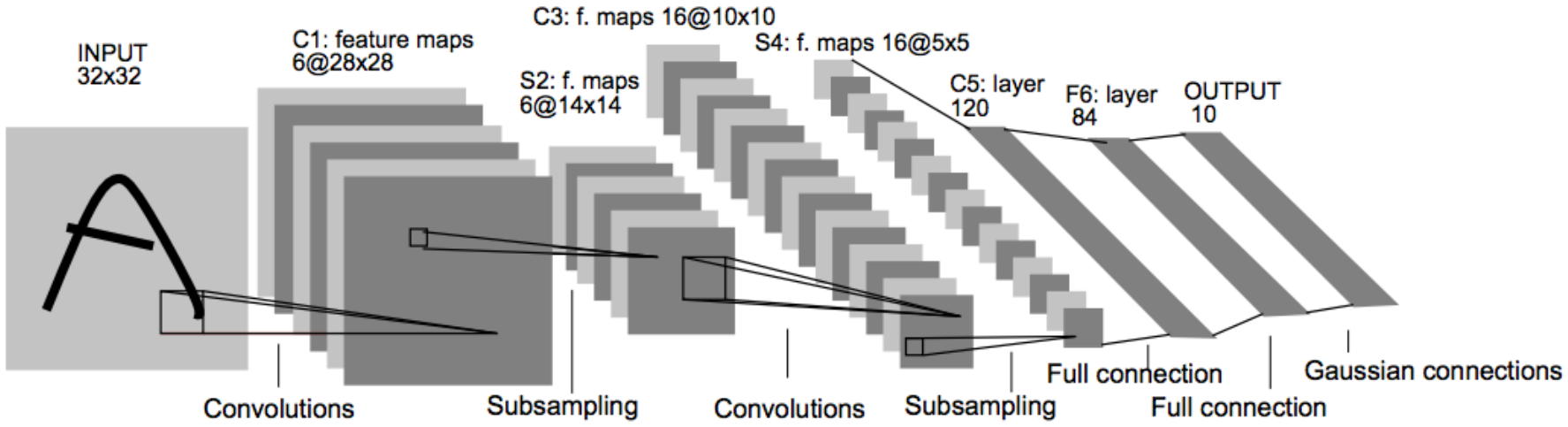
Base

- 19 variáveis (cadastrais + transacionais)
- Uma variável-alvo binária
  - 0 (saudável)
  - 1 (fraudulento)
- 2.943 observações:
  - 1.077 (37%) clientes saudáveis
  - 1.866 (63%) clientes fraudulentos
- Com a opção de pegar apenas clientes analisados pela operação, o problema torna-se mais bem balanceado que o problema original.



Etapas

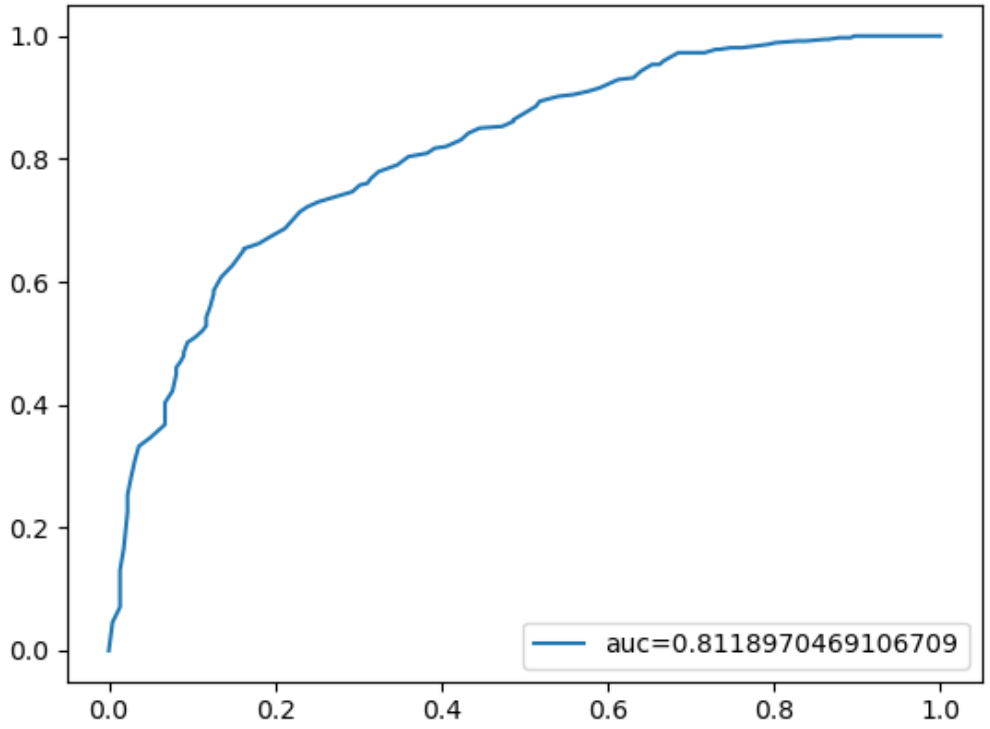
- Uso de Random Forest nos mesmos dados como modelo re referência.
- Geração de imagens a partir dos dados
- Aplicação de CNN como modelo de classificação (LeNet-5).



LeNet-5 Architecture (LeCun et al., 1998)

Resultados

- Random Forest
  - Acurácia Média: 74%
  - Precisão Média: 77%
  - Recall Médio: 84%

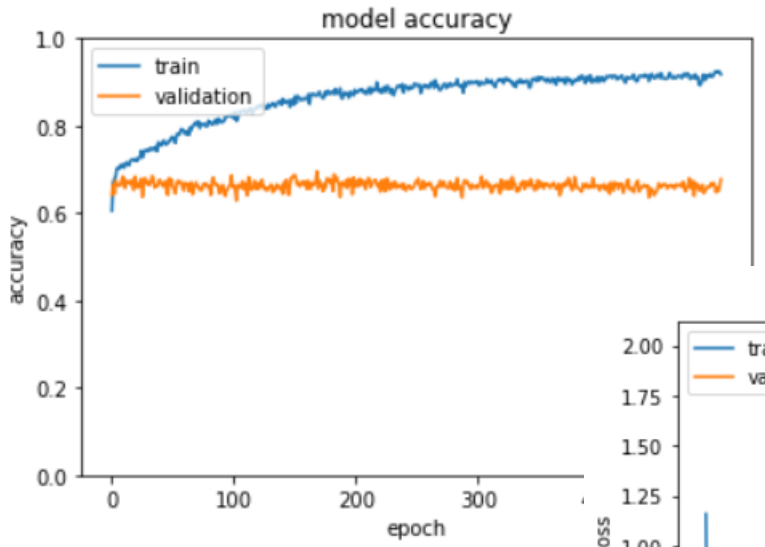


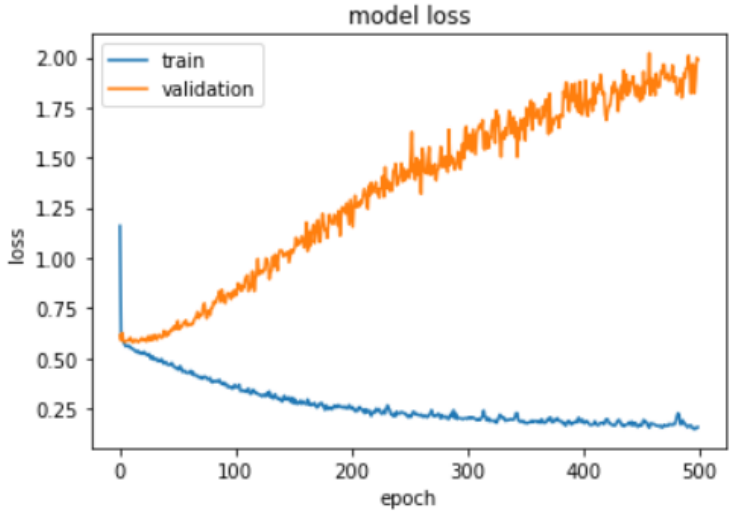
Resultados

- CNN LeNet-5 (500 epochs)
  - vanilla

➢ Acurácia Treino: 92%

➢ Acurácia Validação: 70%



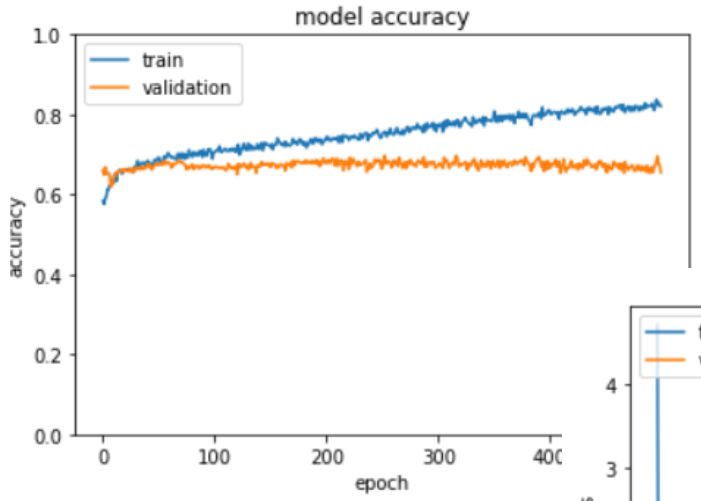


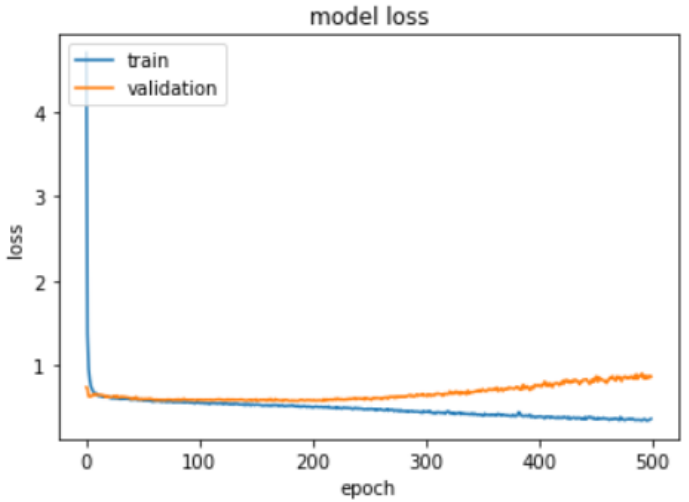

```
Model: "LeNet-5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 36, 36, 6)         60
_________________________________________________________________
average_pooling2d (AveragePo (None, 18, 18, 6)         0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 16, 16, 16)        880
_________________________________________________________________
average_pooling2d_1 (Average (None, 8, 8, 16)          0
_________________________________________________________________
flatten (Flatten)            (None, 1024)              0
_________________________________________________________________
dense (Dense)                (None, 120)               123000
_________________________________________________________________
dense_1 (Dense)              (None, 84)                10164
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 85
=================================================================
Total params: 134,189
Trainable params: 134,189
Non-trainable params: 0
```

Resultados

- CNN LeNet-5 (500 epochs)
  - Inclusão de Dropouts
  - Alteração na quantidade de filtros (6 para 8) na primeira convolução)

➢ Acurácia Treino: 84%
➢ Acurácia Validação: 70%





Model: "LeNet-5"

Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 36, 36, 8)	80
average_pooling2d_2 (Average	(None, 18, 18, 8)	0
dropout (Dropout)	(None, 18, 18, 8)	0
conv2d_3 (Conv2D)	(None, 16, 16, 16)	1168
average_pooling2d_3 (Average	(None, 8, 8, 16)	0
dropout_1 (Dropout)	(None, 8, 8, 16)	0
flatten_1 (Flatten)	(None, 1024)	0
dense_3 (Dense)	(None, 120)	123000
dense_4 (Dense)	(None, 84)	10164
dense_5 (Dense)	(None, 1)	85

Total params: 134,497
Trainable params: 134,497
Non-trainable params: 0

Evoluções futuras e conclusão

- Ajustes na modelagem devido ao overfitting.
- Aplicação de outras configurações de CNNs.
- Se desejarmos uma abordagem considerando comportamento temporal, vale a pena tentarmos RNNs?

- **No caso de minimização do overfitting, a abordagem por CNNs pode se provar mais assertiva que abordagens clássicas.**