Universidade Federal de São Carlos
Centro de Ciências Exatas e Tecnologia
Departamento de Estatística

Relatório - Trabalho de Graduação 2

Detecção de fraude em transações de cartão de crédito via *Machine Learning*

Aluno: Maurício Najjar da Silveira
Orientador: Prof. Dr. Adriano Polpo de Campos

São Carlos, 2018

Sumário

Capítulo 1

Introdução

De forma genérica, uma fraude corresponde a um ato ilícito ou de má fé realizado para se obter vantagens. Segundo um estudo de 2018 da empresa PwC (Lavion, 2018), por volta de 49% das organizações já lidaram com crimes econômicos. Desta forma o que já foi considerado um caso isolado de problema de compliance, hoje é visto com maior evidência pela escala e impacto que a fraude pode causar.

Além do motivo associado à perda financeira, também há a motivação ética de não financiar o terrorismo que é retroalimentado por alguns gêneros de fraude como lavagem de dinheiro. Há inclusive diretrizes vindas do comitê de Basileia para auxiliar o bom gerenciamento dos riscos de fraude on Banking Supervision (2017). Objetivando prevenir estes atos, surge o problema de detecção de fraude por modelos de previsão evitando maiores perdas.

Neste trabalho, será abordado uma classe específica destes atos, fraudes em cartão de crédito, onde o dono do cartão não efetuou as transações pelas quais futuramente será cobrado. Uma das forma de prevenir tais atos reside no controle de quais transações serão aprovadas ou negadas. Portanto é possível tomar a decisão com base no perfil de gastos do cartão, caso se distancie muito, aumentam as chances de ser uma transação fraudulenta.

Um modelo de previsão se faz útil para classificar as transações em fraude ou não,

porém uma das grandes dificuldades é justamente a existência do alto desbalanceamento para a variável resposta. Outro problema de certa forma associado é a proporção de falsos positivos, que pode ser grande caso o modelo de previsão não esteja tão adequado.

O intuito do trabalho é apresentar alternativas de modelos via *Machine Learning* para prever fraude em transações por cartão de crédito. Os objetivos são verificar e compreender a construção de modelos utilizando *Machine Learning* com foco em previsão para detecção de novas transações fraudulentas.

A base de dados obtida é referente a 284.807 transações em cartão de crédito realizadas durante 2 dias em Setembro de 2013 pela Europa. A motivação para um modelo de previsão se encontra na classificação de fraude e não-fraude que é altamente desbalanceada pois as fraudes representam apenas $0,172\%$ do total de transações Falbel (2018b). Este conjunto de dados foi coletado durante uma pesquisa colaborativa entre a empresa Worldline e o grupo de Machine Learning Group da ULB (Université Libre de Bruxelles) (Group, 2018, Dal Pozzolo et al., 2015).

Para detecção do problema foram construídos modelos com 3 enfoques distintos, Rede Neural Artificial, Regressão Logística e Regressão Linear de forma customizada. Ao final são comparados os modelos obtidos no intuito de escolher boas alternativas de detectar automaticamente se uma dada transação constitui como fraudulenta ou não.

Toda manipulação no conjunto de dados, desenvolvimento de modelo e geração de gráficos foi realizado a partir do software livre, R (R Core Team, 2017).

Capítulo 2

Método

Machine Learning é uma área da Inteligência Artificial que faz uso de técnicas estatísticas que permitem ao computador aprender automaticamente com base no conjunto de dados. Wikipedia (2018b). Para o processo de aprendizagem, podem ser considerados métodos supervisionados ou não supervisionados, cuja diferença reside na existência ou não de variável resposta orientando o resultado a ser obtido pelo modelo.

O aprendizado supervisionado permite resolver problemas que possuem p variáveis explicativas $X_i,\ i = 1, \ldots, p$ onde $p \geq 1$ e uma variável resposta Y. O objetivo do modelo é exclusivamente preditivo, consiste na construção de uma função que dependa das variáveis explicativas para prever a variável resposta. Portanto problemas de regressão e classificação fazem parte deste tipo de aprendizado.

O aprendizado não supervisionado é adequado para problemas onde existem p variáveis $X_i,\ i = 1, \ldots, p$ mas nenhuma variável resposta. O objetivo é modelar a estrutura das informações ou sua distribuição conjunta no intuito de aprender mais sobre os dados. Problemas abordados pelo aprendizado não supervisionado são agrupamento e regras de associação.

Resumidamente, métodos supervisionados descrevem modelos onde existe variável resposta para guiar sua construção, portanto o foco é a previsão enquanto métodos não supervisionados se referem a modelos sem variável resposta, o objetivo é identificar

padrões existentes nos dados (Brownlee, 2016).

No problema de detecção de fraude, dado que existe uma resposta binária, ser fraude ou não, o aprendizado mais intuitivo seria o supervisionado, porém algumas construções empregadas realizam a classificação binária praticamente como um objetivo secundário.

Na seção 3, são empregados 8 modelos diferentes que podem ser agrupados em 3 enfoques previamente mencionados:

- Rede Neural Artificial;
- Regressão Logística;
- Regressão Linear.

2.1 Redes Neurais Artificiais

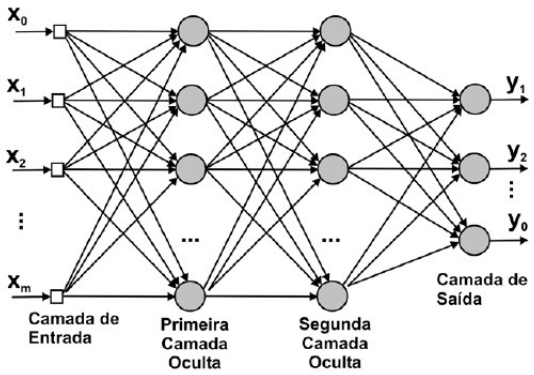

Figura 2.1: Representação visual de uma rede neural artificial

A rede neural artificial é um método de previsão (McCulloch and Pitts, 1943), que corresponde a uma regressão não linear altamente flexível que pode ser representada pela Figura 2.1. Cada neurônio do lado esquerdo representa uma entrada da rede, todos estes neurônios conjuntamente correspondem a camada de entrada. Os neurônios da primeira camada oculta representam uma transformação com base nos neurônios da camada imediatamente anterior. O mesmo vale para a segunda camada oculta,

exclusivamente no caso da camada de saída, as transformações nos neurônios também ocorrem, porém o resultado destas transformações já são as predições da(s) variável(is) resposta (Izbicki and dos Santos, 2018).

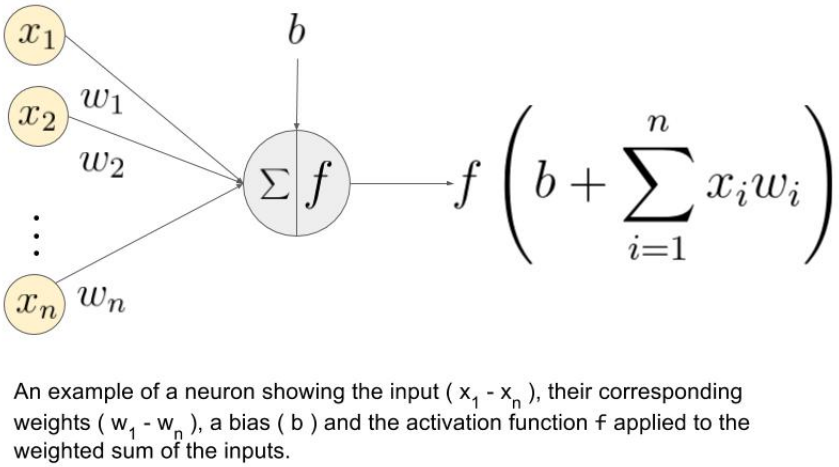

Figura 2.2: Representação visual do cálculo nos neurônios (Sharma, 2017)

A transformação não linear que ocorre em cada neurônio pode ser brevemente descrita como uma combinação linear dos neurônios anteriores aplicados a uma função não linear um-a-um. Adotando como referência a Figura 2.2 para notações, cada neurônio indicado como $x_i, i = 1 \ldots n$ será multiplicado pelo seu peso correspondente w_i e todas estas parcelas serão somadas à constante b. Na soma resultante $b + \sum_{i=1}^{n} x_i w_i$ será aplicada uma função de ativação não linear cujo domínio são os reais e contra domínio o intervalo $]-1; 1[$ para evitar que a estimação iterativa divirja.

O grande objetivo da rede neural é diminuir o erro de previsão minimizando uma função objetivo. A solução para minimizar a função objetivo não é analítica, porém em geral é possível estimar numericamente todos os parâmetros associados a rede neural fixada.

A estimação dos parâmetros é numericamente realizada por gradiente descendente para minimizar a função de perda, que representa a diferença da previsão para a variável

resposta. Em outras palavras, avalia-se o sentido da primeira derivada parcial da função de perda em termos de cada parâmetro. Na iteração seguinte, com os parâmetros atualizados, recalculam-se os gradientes, obtendo ao final do processo iterativo, ao menos um mínimo local onde a derivada parcial será zero em todos os parâmetros. O fato de ser apenas um mínimo local, implica que os parâmetros do modelo dependem substantivamente do chute inicial.

Em especial para redes neurais que possuem sucessivas camadas de neurônios, o método de estimação mais comumente utilizado se denomina Backpropagation, que deriva da estimação por gradiente descendente. A grande diferença do Backpropagation nas redes neurais é que, em cada iteração, é possível alterar apenas os parâmetros da última camada, depois da penúltima e assim sucessivamente. Esta particularidade se verifica pela própria construção hierárquica da rede.

Devido ao método de estimação, é necessário que as funções de ativação sejam deriváveis, assim basta aplicar regras da cadeia para o cálculo dos gradientes por parâmetro.

As implementações de redes neurais foram possíveis por meio do pacote 'Keras', original da linguagem Python. (Allaire and Chollet)

2.2 AutoEncoder

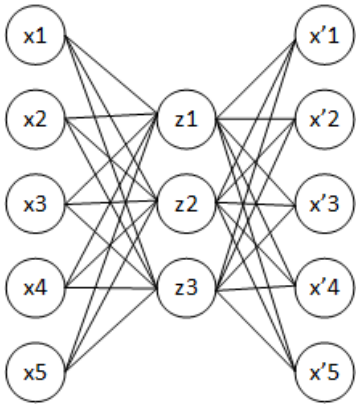

Figura 2.3: Representação visual do formato de um AutoEncoder (Programmer, 2015)

Um AutoEncoder é uma rede neural treinada para tentar copiar sua camada de entrada para a camada de saída. Como exemplificado na Figura 2.3, a construção da rede consiste em 2 etapas: uma função $\mathbf{h} = \mathbf{f}(\mathbf{x})$ que codifica as variáveis de entrada e outra função $\mathbf{r} = \mathbf{g}(\mathbf{h})$ que decodifica e tenta reconstruir as variáveis de entrada.

À primeira vista, copiar as entradas não parece fornecer nenhuma informação relevante, porém a arquitetura da rede restringe que seja aprendida uma função identidade que reproduza perfeitamente as variáveis de entrada. Esta restrição está implícita pela presença do bottleneck, que consiste em uma camada oculta com número de neurônios menor que a quantidade de variáveis de entrada/saída.

A sua aplicação mais conhecida é na redução de dimensionalidade pois na etapa da codificação, a função $\mathbf{h}$ comprime p variáveis em d neurônios e na decodificação expande os valores dos d neurônios em p neurônios na camada de saída. A camada com estes d neurônios reúne grande parte das informações de forma condensada, portanto é vista como uma alternativa à análise de componentes principais, só não possui a interpretabilidade intuitiva proveniente deste método.

Normalmente, AutoEncoders são treinados com apenas uma camada de codificação (Encoder) e uma camada de decodificação (Decoder), porém é possível inserir mais camadas, dando maior complexidade ao modelo. Esta maior profundidade da rede flexibiliza os padrões a serem encontrados.

Detecção de Anomalias via AutoEncoder

Em geral o AutoEncoder é utilizado para aprendizado não supervisionado pois o objetivo é encontrar padrões no conjunto de dados. Dado que para detectar fraudes o intuito é previsão, o mais intuitivo seria partir para um método de aprendizado supervisionado, portanto apenas o AutoEncoder não resolve a solução para o problema (Goodfellow et al., 2016).

Este elegante método para o problema proposto para classificar em fraude e não fraude provém do artigo de Schreyer et al. (2017) onde se faz uso de aprendizado não

supervisionado, AutoEncoder, treinando o modelo apenas para dados de não fraude de modo que o erro de previsão se torna um classificador de fraude.

Para compreender a intuição por trás deste método, é necessário supor que a distribuição conjunta das covariáveis difere entre fraude e não fraude. Desta forma os padrões identificados pelo AutoEncoder em transações não fraudulentas, não estarão presentes em transações fraudulentas, logo terão maior erro de previsão no AutoEncoder. A partir do erro de previsão, basta determinar um ponto de corte k no qual a partir deste, todas as transações com erro de previsão maior que k serão classificadas como fraude (Falbel, 2018a). Em suma, o método pode ser renomeado como One-Class Classification pois define-se uma classe e o resto se classifica como uma segunda categoria.

2.3 Regressão Logística

Criada pelo estatístico David Cox em 1958, a regressão logística é um famoso modelo estatístico utilizado principalmente para modelar respostas binárias, por meio da probabilidade de ocorrência de um evento. Formalmente, a regressão linear logística em sua forma mais simples corresponde a um modelo linear generalizado para a distribuição de probabilidade bernoulli com função de ligação logito. (Walker and Duncan, 1967)

Considerando Y binária, com distribuição Bernoulli de esperança π. Logo

$$Y_i \ = \ \pi_i + \epsilon_i, \ i = 1, \ldots, n$$

onde

$$\pi_i \ = \ \frac{exp\{\eta_i\}}{1 + exp\{\eta_i\}}, \ i = 1, \ldots, n$$

$$\eta_i \ = \ \beta_0 + \beta_1 X_{i1} \cdots + \beta_p X_{ip}, \ i = 1, \ldots, n$$

A estimação dos parâmetros é feita minimizando a função de máxima verossimilhança numericamente por uma pequena adaptação da minimização por Newton-Raphson.

Além da função de ligação logito

$$\eta_i \;=\; log\;\frac{\pi_i}{1-\pi_u}$$

também existem outras funções de ligação aplicáveis, como a C-Log-Log

$$\eta_i \;=\; log(-log(1-\pi_i))$$

que foi utilizada neste trabalho na forma de uma alternativa para adequar o desbalanceamento dos dados.

2.4 Penalização Lasso

A penalização Lasso é um método que visa encontrar um estimador de regressão melhor que o encontrado por mínimos quadrados em termos do risco. Isso ocorre pela inserção de um termo a mais na função de risco, que penaliza a existência de cada parâmetro diferente de 0 (Izbicki and dos Santos, 2018). A função para estimar $\boldsymbol{\beta}$ pode ser obtido minimizando a equação

$$\hat{\boldsymbol{\beta}}_{L_1,\lambda} \;=\; arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} \Big(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{i,j}\Big)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

Desta forma, por métodos numéricos estimam-se diferentes valores de λ, que produzem um risco diferente na base de validação, quanto menor este risco, mais adequado está o valor do λ.

Capítulo 3

Resultados

Antes de iniciar qualquer análise é importante anunciar que as covariáveis originais foram descaracterizadas por componentes principais para manter a confidencialidade dos clientes. Isso implica que as variáveis V1 a V28 são ortogonais entre si, mas não ortonormais, o que compromete a interpretabilidade do modelo mas tem a capacidade de gerar bons modelos preditivos. Também há a variável Tempo que registra o horário em segundos desde a primeira transação da base, o que pode ser traduzido no máximo no horário da transação em cada um dos 2 dias do conjunto de dados. A variável Valor, só não foi descaracterizada conjuntamente com as demais pois faz-se importante o seu uso na escolha de um eventual ponto de corte sensitivo ao Valor de fraude prevenido.

3.1 Análise Descritiva

Para entender de forma preliminar a distribuição de cada covariável condicional a sua classificação, foi gerada a Figura 3.1. Todas as variáveis foram padronizadas no intervalo $[0, 1]$ para possibilitar a visualização em um único gráfico.

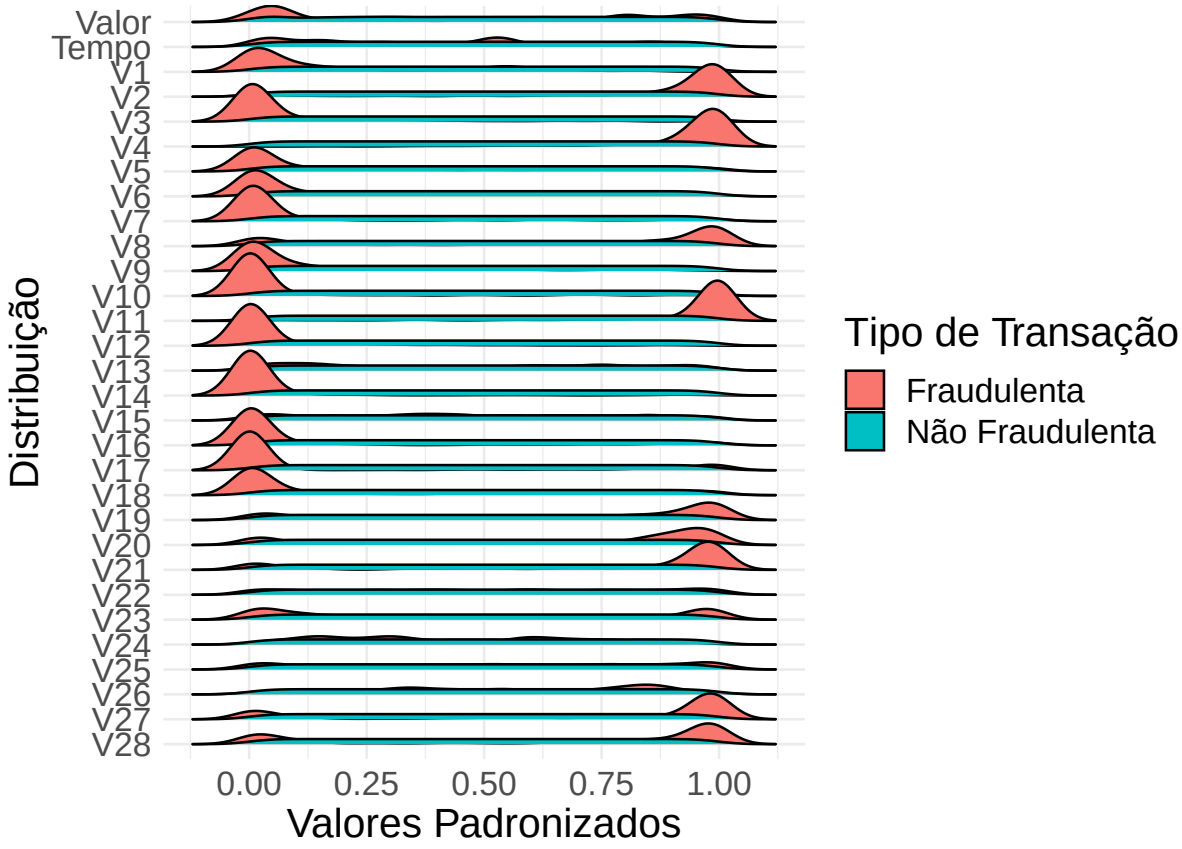

Figura 3.1: Distribuição das covariáveis por tipo de transação

A partir da Figura 3.1, é possível afirmar que em muitos componentes, são claramente distintas as distribuições condicionais na variável resposta. Esta evidente distinção que permite a construção de modelos capazes de detectar uma transação fraudulenta.

Vale comentar que os componentes, possuem distribuições ligeiramente menos discrepantes a medida que o componente possui maior índice, esta característica deve derivar do fato de que os primeiros componentes explicam maior variabilidade que os últimos.

3.2 Rede Neural Artificial

Para inserir as covariáveis na RNA, houveram manipulações no conjunto de dados com o objetivo de diminuir a escala. A transformação aplicada foi a seguinte para cada covariável:

$$z = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Esta transformação leva o domínio de cada covariável ao contra domínio $[0, 1]$, de modo a dificultar a divergência do método iterativo de estimação dos parâmetros.

3.2.1 Modelo 1 - AutoEncoder Preditivo

O modelo de AutoEncoder, que equivale a uma rede neural genérica, possui 30 variáveis de entrada (28 Componentes, Valor e Tempo), 15 neurônios na primeira camada oculta, 10 neurônios na segunda camada oculta, 15 neurônios na terceira camada oculta e 30 saídas (associadas às variáveis de entrada).

A camada de entrada juntamente com a primeira e a segunda camada oculta, caracterizam o Encoder que resume informações, a segunda e a terceira camada oculta juntamente com a camada de saída caracterizam o Decoder que decodifica os valores presentes na segunda camada oculta. Todas as camadas são densas, ou seja, todos os neurônios da camada em questão recebem todos os neurônios da camada anterior. A função de ativação de todas as camadas foi a Tangente Hiperbólica (3.1) por ter como domínio os reais e contra domínio o intervalo de $[-1, 1]$, no intuito de estabelecer limites e evitar que o método numérico possa divergir.

$$tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{3.1}$$

Ao construir o modelo acima descrito, são necessários 1270 parâmetros ao todo, distribuídos segundo a Tabela 3.1.

Tabela 3.1: Distribuição dos Parâmetros na Rede Neural

Camada	**# Neurônios**	**# Parâmetros**
Entrada	30	-
Camada Oculta 1	15	465
Camada Oculta 2	10	160
Camada Oculta 3	15	165
Saída	30	480
Total	-	1270

Para estimar os parâmetros, o modelo foi treinado unicamente com observações associadas a transações não fraudulentas que correspondem a mais de 99% do conjunto de dados.

O erro de previsão foi calculado como o Erro Quadrático Médio por observação, desta forma é possível avaliar em uma medida única por observação o quanto as covariáveis se distanciaram dos valores preditos.

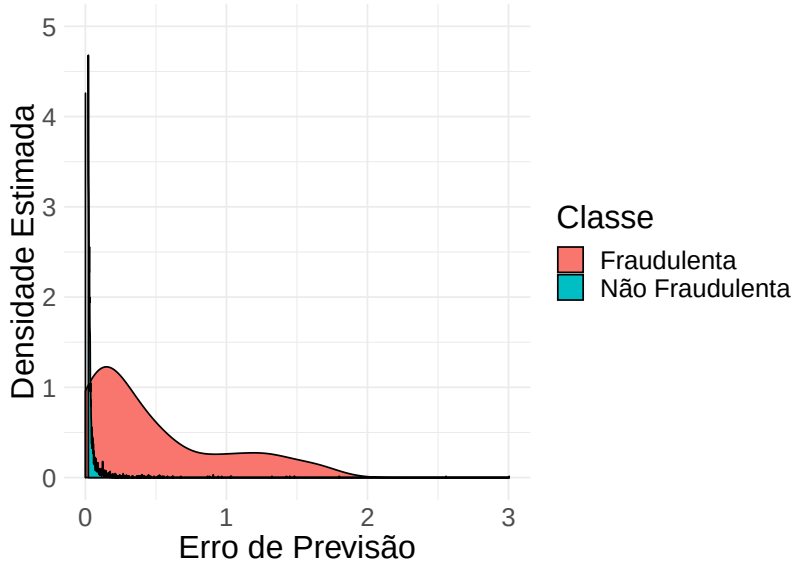

Figura 3.2: Distribuição do Erro de Previsão por tipo de transação

A Figura 3.2 indica que o erro de previsão para transações não fraudulentas possui

uma função de densidade concentrada em torno de $0,1$, enquanto a distribuição do erro de previsão para transações não fraudulentas varia entre 0 e 2. Desta forma, o comportamento observado no gráfico demonstra que o modelo de fato capturou os padrões inerentes das transações não fraudulentas e as que se distanciam deste perfil em geral são fraude.

No intuito de determinar um ponto de corte k a partir do Erro de Previsão associado a i-ésima observação, $Erro_i$, será considerada a forma abaixo.

$$Classe_i = \begin{cases} \text{Não Fraude}, & \text{se } Erro_i \leq k \\ \text{Fraude}, & \text{CC} \end{cases}$$

3.2.2 Modelo 2 - AutoEncoder Resposta Binária

O modelo de AutoEncoder com resposta binária, corresponde a uma rede neural semelhante ao do AutoEncoder utilizado no modelo 1 porém com uma única saída de resposta binaria. A estrutura possui 30 variáveis de entrada (28 Componentes, Valor e Tempo), 15 neurônios na primeira camada oculta, 10 neurônios na segunda camada oculta, 15 neurônios na terceira camada oculta e 1 saída associada a variável resposta.

Todas as camadas são densas, ou seja, todos os neurônios da camada em questão recebem todos os neurônios da camada anterior. A função de ativação das 3 primeiras camadas foi a Tangente Hiperbólica (3.1) enquanto a saída teve como função de ativação a função sigmóide/logito que gera resposta no intervalo $[0,1]$ ideal para as respostas 0 e 1. Para a função de perda, foi utilizado a Cross-Entropy Binária que dificulta o aprendizado de apenas uma das classes mesmo que seja predominante.

Ao construir o modelo acima descrito, são necessários 1301 parâmetros ao todo, distribuídos da segundo a Tabela 3.2.

Tabela 3.2: Distribuição dos Parâmetros na Rede Neural

Camada	**# Neurônios**	**# Parâmetros**
Entrada	30	-
Camada Oculta 1	15	465
Camada Oculta 2	10	160
Camada Oculta 3	15	165
Camada Oculta 4	30	480
Saída	1	31
Total	-	1301

Diferentemente do primeiro modelo, o modelo 2 foi treinado com todas as observações do conjunto de treino/validação.

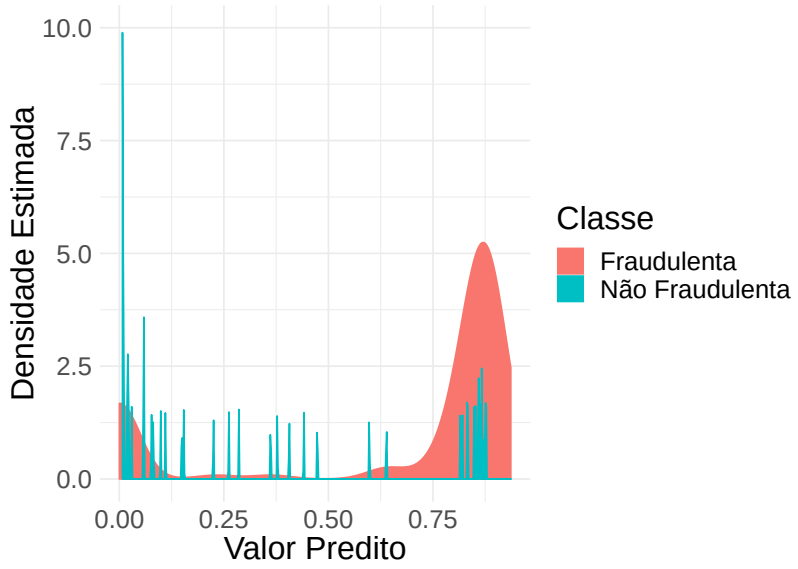

Figura 3.3: Distribuição da Previsão por tipo de transação

A Figura 3.3, ilustra o comportamento da distribuição do valor predito para as transações de cada classe, notavelmente o modelo aprendeu o que caracteriza como não fraude devido a alta concentração próxima do zero. Quanto às transações de fraude, não estão perfeitamente no 1, mas a maior densidade está apartada do zero,

este comportamento é ideal para futuramente adotar um ponto de corte.

No intuito de determinar um ponto de corte k a partir do valor predito associado a i-ésima observação, $\hat{Y}_i$, será considerada a forma abaixo.

$$Classe_i = \begin{cases} \text{Não Fraude}, & \text{se } \hat{Y}_i \leq k \\ \text{Fraude}, & \text{CC} \end{cases}$$

3.2.3 Modelo 3 - RNA Binária

O modelo de Rede Neural Artificial com resposta binária, corresponde a construção de rede neural mais comum com uma única saída de resposta binaria. A estrutura possui 30 variáveis de entrada (28 Componentes, Valor e Tempo), 15 neurônios na primeira camada oculta, 15 neurônios na segunda camada oculta, 15 neurônios na terceira camada oculta e 1 saída associada a variável resposta binária.

Todas as camadas são densas, ou seja, todos os neurônios da camada em questão recebem todos os neurônios da camada anterior. A função de ativação das 3 primeiras camadas foi a ReLU (Rectified Linear Unit) cuja função é expressa em 3.2, para o neurônio de saída, foi utilizada a função de ligação sigmóide/logito que gera resposta no intervalo $[0, 1]$ ideal para as respostas 0 e 1. Para a função de perda, foi utilizado a Cross-Entropy Binária que dificulta o aprendizado de apenas uma das classes mesmo que esta seja predominante.

$$\max(0, x) \tag{3.2}$$

Ao construir o modelo acima descrito, são necessários 961 parâmetros ao todo, distribuídos da segundo a Tabela 3.3.

Tabela 3.3: Distribuição dos Parâmetros na Rede Neural

Camada	**# Neurônios**	**# Parâmetros**
Entrada	30	-
Camada Oculta 1	15	465
Camada Oculta 2	15	240
Camada Oculta 3	15	240
Saída	1	16
Total	-	961

Diferentemente do primeiro modelo, o modelo 3 foi treinado com todas as observações do conjunto de treino/validação.

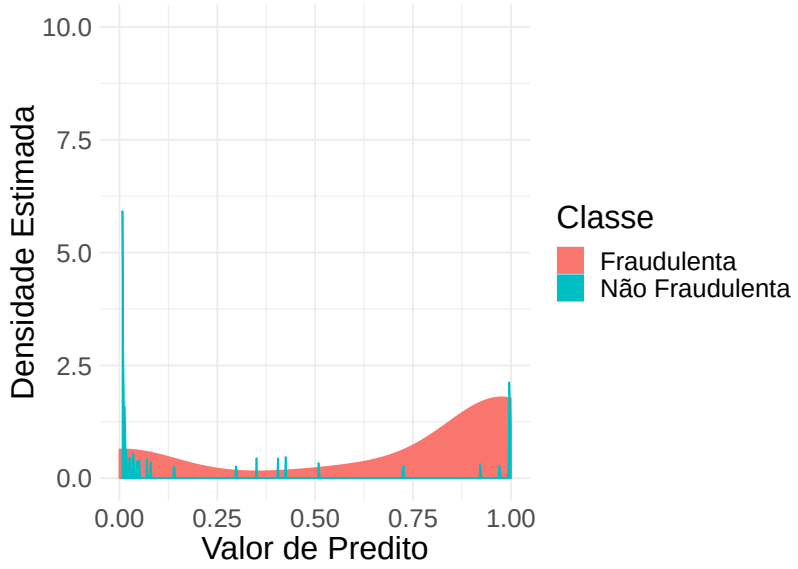

Figura 3.4: Distribuição da Previsão por tipo de transação

A Figura 3.4, ilustra o comportamento da distribuição do valor predito para as transações de cada classe, é importante apenas avisar que sem limitar o tamanho do eixo y do gráfico, a densidade estimada da não fraude no 0 explodia bem mais do que 10. Quanto ao resto, o modelo aprendeu bem sobre o perfil da fraude, que apesar de ficar mais distribuído, é uma característica ideal para uma medida de risco/score.

No intuito de determinar um ponto de corte k a partir do valor predito associado a i-ésima observação, $\hat{Y}_i$, será considerada a forma abaixo.

$$Classe_i = \begin{cases} \text{Não Fraude}, & \text{se } \hat{Y}_i \leq k \\ \text{Fraude}, & \text{CC} \end{cases}$$

3.3 Regressão Logística

Não foram realizadas manipulações no conjunto de dados para os modelos que seguem a partir da regressão logística.

Nos modelos 4, 5 e 6 que seguem, por se tratarem de derivações de uma regressão logística, terão sua posterior classificação da forma:

$$Classe_i = \begin{cases} \text{Fraude}, & \text{se } \hat{Y}_i \geq k \\ \text{Não Fraude}, & \text{CC} \end{cases}$$

3.3.1 Modelo 4 - Logística Completa

O modelo 4 representa a regressão logística múltipla da indicadora de fraude em todas as covariáveis. A visualização da eficácia do modelo estará visível no Capítulo 4 incluindo curva ROC e algumas demais medidas que avaliam o desempenho do modelo.

3.3.2 Modelo 5 - Logística Completa ligação C-Log-Log

O modelo 5 representa a regressão logística múltipla da indicadora de fraude em todas as covariáveis utilizando a função de ligação C-Log-Log por ajustar bem a assimetria.

3.3.3 Modelo 6 - Lasso Logística

O modelo 6 representa uma regressão logística múltipla com penalização Lasso. Possivelmente devido a assimetria ou a independência entre a maioria das covariáveis, o

método levou muito mais tempo, praticamente diverge e encontra valores altos para λ, o que leva a serem removidos todos os parâmetros.

3.4 Regressão Linear

No intuito de construir modelos via regressão linear de forma personalizada ao problema de detecção de transações fraudulentas, foi utilizada a seguinte transformação na variável resposta.

$$Y_i^* = \begin{cases} (0,01) * Valor_i, \ \textit{se transação i não for fraudulenta} \\ (-1) * Valor_i, \ CC \end{cases}$$

Esta transformação tem o objetivo de representar o custo financeiro para uma instituição que ressarce o valor completo de todas as fraudes e recebe 1 % do valor de toda transação não fraudulenta. Esta ordenação, de quanto menor o valor predito maior o risco de fraude, representa uma forma distinta de visualizar o problema que pode auxiliar a prevenir a fraude em valor absoluto.

No intuito de determinar um ponto de corte k a partir do $\hat{Y}^*$ predito, nos modelos 7 e 8 que seguem, será considerada a forma abaixo para posterior classificação.

$$Classe_i = \begin{cases} \text{Fraude}, & \text{se } \hat{Y}_i^* \leq k \\ \text{Não Fraude}, & \text{CC} \end{cases}$$

3.4.1 Modelo 7 - Lasso Linear

O modelo 7 representa a regressão linear da variável transformada Y^* em um subconjunto das covariáveis excetuando a variável Valor. Especificamente para este modelo, devido a penalização Lasso, as covariáveis automaticamente removidas do modelo por não compensarem a redução do risco foram Tempo, V13, V15, V21, V23, V24, V25, V26 e V28. O fato de terem sido removidas covariáveis de maior número, está coerente

com a origem das covariáveis por componentes principais, pois estas não agregam tanto em variabilidade.

O método de treinamento por validação cruzada foi o K-Fold com $k = 7$, a medida de risco foi o erro absoluto médio, para selecionar o λ ideal, foi escolhido o λ que gerava menor risco acrescido de 1 desvio padrão, o que gera um lambda um pouco maior do que o de menor risco.

3.4.2 Modelo 8 - Linear Completa

O modelo 8 representa a regressão linear múltipla da variável transformada Y^* em todas as covariáveis. Importante salientar que a variável Valor não foi incluída no modelo.

Capítulo 4

Comparação dos Modelos

Antes de comparar a performance entre os modelos, é imprescindível comentar que foram utilizadas partições diferentes do banco de dados para treinar o modelo, validá-lo, estabelecer o ponto de corte e posteriormente usá-lo como teste.

Para facilitar a compreensão dos gráficos a seguir, segue um pequeno resumo do que constitui cada um dos 8 modelos apresentados.

- Modelo 1: One-Class Classification via AutoEncoder;
- Modelo 2: AutoEncoder com resposta binária;
- Modelo 3: RNA comum com resposta binária;
- Modelo 4: Regressão Logística Completa ligação Logito;
- Modelo 5: Regressão Logística Completa ligação C-Log-Log;
- Modelo 6: Regressão Lasso Logística;
- Modelo 7: Regressão Lasso Linear;
- Modelo 8: Regressão Linear Completa.

4.1 Curva ROC

Para construir a curva ROC, quase ironicamente, foi considerada a fraude como evento de sucesso (1) e a não fraude como fracasso (0). Desta forma as medidas de Sensibilidade e Especificidade foram calculadas como:

- Sensibilidade: $P(\hat{Y} = 1 \mid Y = 1)$
- Especificidade: $P(\hat{Y} = 0 \mid Y = 0)$

Antes de apresentar a curva ROC de cada modelo, vale lembrar que neste caso, como o evento de fraude é raro, especificidade alta não é uma medida tão informativa dado que um modelo bobo poderia apenas classificar aleatoriamente 99% das transações como não fraude e teria uma boa especificidade.

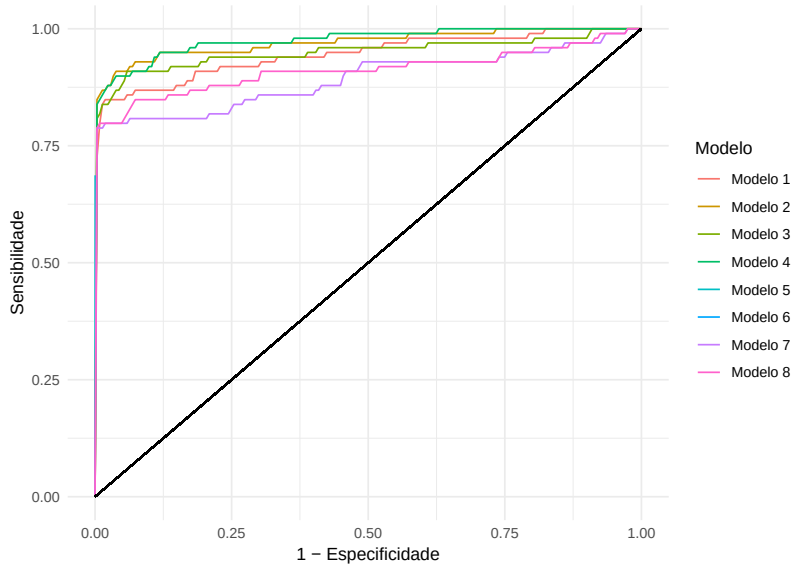

Figura 4.1: Curva ROC dos modelos 1 a 8

A partir da Figura 4.1, observa-se que os modelos 7 e 8 apresentaram um desempenho abaixo dos demais em termos das medidas sensibilidade e especificidade. Outro ponto a comentar é a inexistência dos modelos 5 e 6, que correspondem respectivamente aos modelos, C-Log-Log e Lasso-Logístico, pois as predições ficaram muito próximas

entre si. Exclusivamente o modelo 6 (Lasso-Logístico), dado que todos os parâmetros foram removidos,ficou apenas o intercepto que não gera variabilidade ao modelo impossibilitando que constitua um modelo de classificação útil. Os modelos 2 e 4 obtiveram o melhor desempenho seguidos dos modelos 1 e 3.

No entanto, a curva ROC não permite interpretar o valor de fraude recuperado em cada ponto de corte, nem a proporção de transações não fraudulentas negadas para cada fraude prevenida. Nos 3 próximos gráficos, dividindo em 4 cenários, negando $0.5\%, 1\%, 2\% ou 5\%$ das transações de maior risco de fraude, é possível avaliar quais modelos geram melhores resultados monetariamente e de eficiência.

Para estas medidas a seguir, foram removidos os modelos 5 e 6 que não geraram variabilidade suficiente, seus valores não permitem a detecção de transações fraudulentas de forma inteligente.

4.2 Penetração

A medida penetração pode ser vista com a mesma fórmula da sensibilidade pois mensura o percentual das fraudes que o modelo conseguiu identificar.

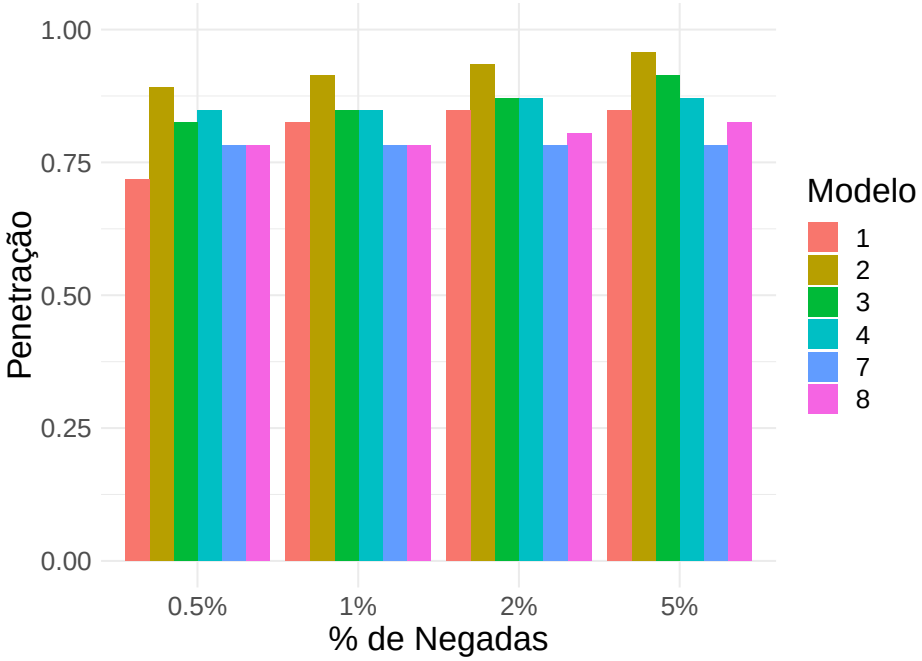

Figura 4.2: Comparação da penetração para cada modelo

Assim como foi observado na curva ROC, na Figura 4.2 nota-se que no cenário mais conservador (negando 0.5 %), o modelo 1, One-Class Classification, possui a menor penetração, identificando apenas pouco mais de 70% das fraudes. Consistentemente o modelo 2 gera uma penetração acima de 87.5% o que já sugere ser o melhor modelo desenvolvido. Os modelos 7 e 8 que não tinham gerado uma curva ROC tão favorável não aparenta gerar uma performance muito aquém dos demais modelos. A Logística completa, representada pelo modelo 4, teve uma performance semelhante ao modelo 3 (RNA resposta binária).

4.3 Hit Rate

A medida Hit Rate pode ser vista como o percentual de fraudes em relação ao que foi classificado como fraude.

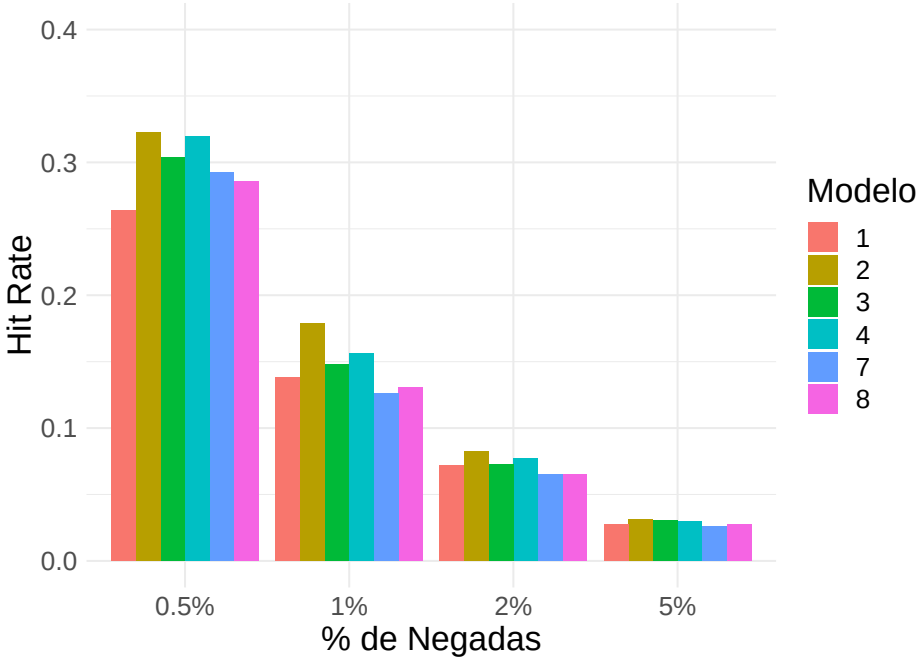

Figura 4.3: Comparação do hit rate para cada modelo

Diferentemente da Figura 4.2, no gráfico da Figura 4.3 há uma sensível diferença entre cenários de percentual de transações a serem negadas. Quanto maior o percentual de transações a serem negadas, menor o Hit Rate. A interpretação desta medida para o exemplo do modelo 4 no cenário de negar 1% das transações, é que dentre as transações classificadas como fraude pelo modelo, apenas 15% são de fato fraude.

Nesta Figura 4.3, observa-se que novamente o modelo com melhor desempenho é o modelo 2, mas desta vez seguido do Logístico completo representado pelo modelo 4. O pior modelo para esta medida é o 1 (One-Class Classification).

4.4 Valor Prevenido

A medida 'valor prevenido de fraude' mensura em termos monetários o valor evitado em transações fraudulentas.

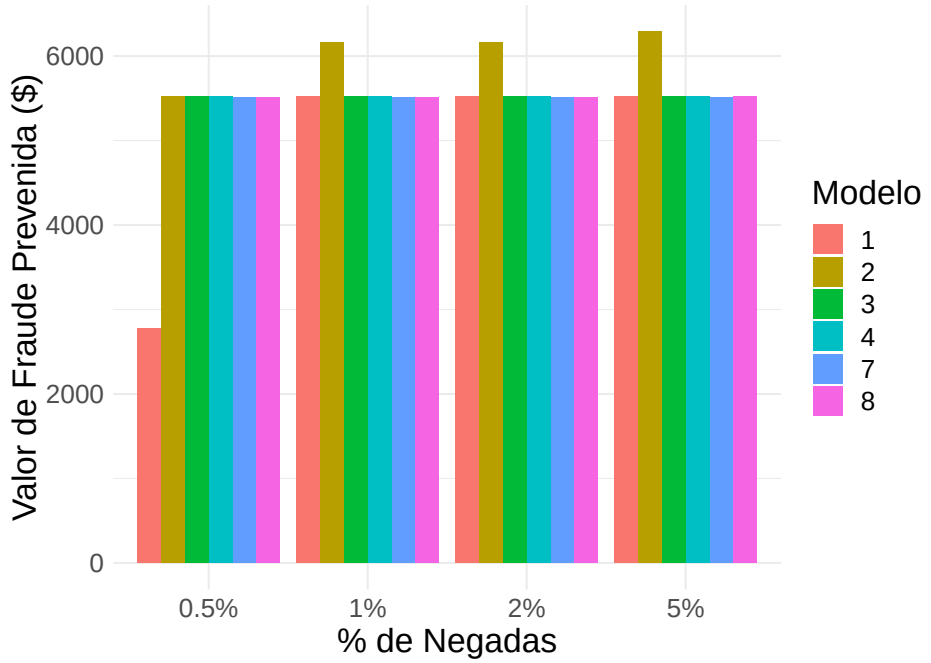

Figura 4.4: Comparação do valor prevenido de fraude para cada modelo

Como se observa na Figura 4.4, nesta última medida, há apenas 4 barras que diferem das demais, estas são do modelo 1 no cenário mais conservador e do modelo 2 nos 3 outros cenários. Vale comentar que mesmo os modelos 7 e 8 tendo desempenho relativamente pior que os demais, não se distanciou de forma tão considerável nestes 3 gráficos de medidas de eficiência.

Capítulo 5

Comentários Finais

Em uma primeira análise, a partir dos gráficos presentes nas Figuras 3.2, 3.3 e 3.4 é evidente que algumas transações são erroneamente classificadas como fraude. Essa característica é esperada dado que nenhum modelo será perfeito, porém revela que estas transações possuem perfil de fraude, o que justifica negar o cliente bom devido ao alto risco.

Finalizadas as comparações dos modelos, se fosse necessário optar por um único método este seria o modelo 2 que gerou bons indicadores, porém excetuando os modelos 5 e 6, todos obtiveram uma performance considerável. O modelo Lasso-Logístico não convergiu para um λ adequado, possivelmente devido ao alto desbalanceamento e a falta de ganho pelo uso majoritário de covariáveis ortogonais. O modelo Logístico com função de ligação C-Log-Log gerou também uma baixíssima variabilidade por mais que tenha ajustado bem boa parte dos zeros devido ao alto desbalanceamento.

Os modelos 7 e 8 que representavam a regressão linear na variável transformada, apesar de não terem performado tão bem nas medidas da curva ROC, gerou um desempenho similar aos demais modelos no que tange a valor prevenido de fraude. Os modelos 1, 2 e 3 provenientes de RNA por diferentes formatos, teve uma performance muito boa também, principalmente comparado ao modelo Logístico completo que no primeiro trabalho aparentava ser ligeiramente melhor que o One-Class Classification.

De qualquer forma não é tão notável essa distinção entre os modelos, apenas o modelo 2 que consistentemente gerou indicadores melhores, o que é algo inesperado devido à construção diferente de AutoEncoder com uma única camada de saída binária.

Em relação ao conjunto de dados, vale uma pequena crítica, pois existe a ilusão de independência entre observações/transações, quando na verdade se uma transação é fraude, a próxima do mesmo cartão tem uma boa probabilidade de também ser. Imagina-se que estas considerações de efeito temporal estejam presentes de forma oculta nos componentes principais, porém claramente seria preferível trabalhar com as variáveis originais, a interpretabilidade sai bem prejudicada por este motivo.

Referências Bibliográficas

JJ Allaire and François Chollet. *keras: R Interface to 'Keras'*. URL `https://keras.rstudio.com`. R package version 2.1.6.9000.

Jason Brownlee. Supervised and unsupervised machine learning algorithms, 2016. `https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/` [Acessado: 21-06-2018].

Andrea Dal Pozzolo, Olivier Caelen, Reid A Johnson, and Gianluca Bontempi. Calibrating probability with undersampling for unbalanced classification. In *Computational Intelligence, 2015 IEEE Symposium Series on*, pages 159–166. IEEE, 2015.

Daniel Falbel. Predicting fraud with autoencoders and keras, 2018a. `https://tensorflow.rstudio.com/blog/keras-fraud-autoencoder.html` [Acessado: 22-06-2018].

Daniel Falbel. Predicting fraud with autoencoders and keras, 2018b. `https://www.kaggle.com/mlg-ulb/creditcardfraud` [Acessado: 22-06-2018].

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

Machine Learning Group. Machine learning group, 2018. `http://mlg.ulb.ac.be/` [Acessado: 07-06-2018].

Rafael Izbicki and Tiago Mendonça dos Santos. *Machine Learning sob a ótica estatística*. Notas de Aula, 2018.

Didier Lavion. Pulling fraud out of the shadows, 2018. `https://www.pwc.com/gx/en/forensics/global-economic-crime-and-fraud-survey-2018.pdf` [Acessado: 21-06-2018].

Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

John Neter, Michael H Kutner, Christopher J Nachtsheim, and William Wasserman. *Applied linear statistical models*, volume 4. Irwin Chicago, 1996.

Basel Committee on Banking Supervision. Sound management of risks related to money laundering and financing of terrorism, 2017. `https://www.bis.org/bcbs/publ/d405.pdf`[Acessado: 14-11-2018].

Lazy Programmer. A tutorial on autoencoders for deep learning - lazy programmer, 2015. `https://www.learnopencv.com/understanding-activation-functions-in-deep-learning/` [Acessado: 22-06-2018].

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2017. URL `https://www.R-project.org/`.

Jurgen Schmidhuber. *Deep Learning in Neural Networks: An Overview*. Technical Report IDSIA, 2014.

Marco Schreyer, Timur Sattarov, Damian Borth, Andreas Dengel, and Bernd Reimer. Detection of anomalies in large scale accounting data using deep autoencoder networks. *arXiv preprint arXiv:1709.05254*, 2017.

Aditya Sharma. Understanding activation functions in deep learning learn opencv, 2017. `https://www.learnopencv.com/understanding-activation-functions-in-deep-learning/` [Acessado: 21-06-2018].

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

Strother H Walker and David B Duncan. Estimation of the probability of an event as a function of several independent variables. *Biometrika*, 54(1-2):167–179, 1967.

Wikipedia. Data analysis techniques for fraud detection - wikipedia, 2018a. `https://en.wikipedia.org/wiki/Data_analysis_techniques_for_fraud_detection` [Acessado: 21-06-2018].

Wikipedia. Machine learning - wikipedia, 2018b. `https://en.wikipedia.org/wiki/Machine_learning`[Acessado: 21-06-2018].