CENTRO UNIVERSITÁRIO DE ANÁPOLIS – UniEVANGÉLICA
BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO

HUGO FERNANDES VIEIRA

DETECÇÃO DE FRAUDE NO USO DE CARTÕES DE CRÉDITO UTILIZANDO TÉCNICAS SUPERVISIONADAS DE *MACHINE LEARNING*

ANÁPOLIS - GO
2018

HUGO FERNANDES VIEIRA

DETECÇÃO DE FRAUDE NO USO DE CARTÕES DE CRÉDITO UTILIZANDO TÉCNICAS SUPERVISIONADAS DE *MACHINE LEARNING*

Trabalho de Conclusão de Curso I apresentado como requisito parcial para a conclusão da disciplina de Trabalho de Conclusão de Curso I do curso de Bacharelado em Engenharia de Computação do Centro Universitário de Anápolis – UniEVANGÉLICA.

Orientador(a): Prof. Me. Marcelo de Castro Cardoso

Co-orientador(a): Prof. Mª Luciana Nish

Anápolis - GO

2018

HUGO FERNANDES VIEIRA

DETECÇÃO DE FRAUDE NO USO DE CARTÕES DE CRÉDITO UTILIZANDO TÉCNICAS SUPERVISIONADAS DE MACHINE LEARNING

Trabalho de Conclusão de Curso II apresentado como requisito parcial para a obtenção de grau do curso de Bacharelado em Engenharia de Computação do Centro Universitário de Anápolis – UniEVANGÉLICA.

Aprovado(a) pela banca examinadora em 11 de junho de 2019, composta por:

Marcelo de Castro Cardoso
Presidente da Banca

Clarimar José Coelho
Prof(a). Convidado(a)

Nilton Correia da Silva
Prof(a). Convidado(a)

Resumo

A cada ano cresce o uso dos cartões de crédito como forma de pagamento. Isso gera como consequência a atenção dos criminosos que estão em busca de maneiras de fraudar as transações. Para diminuir o prejuízo, empresas prestadoras do serviço de cartão de crédito estão sempre procurando novos métodos para combater esses crimes. O grande problema é que são milhares de transações por minuto e algumas fraudes passam despercebidas. Neste trabalho, será proposto um estudo de técnicas de *machine learning* que possam analisar, compreender e detectar possíveis fraudes. Para que possam aplicar as técnicas que possam alcançar o melhor resultado na detecção de fraudes. E também fazer uma análise comparativa entre trabalhos relacionados ao tema e mostrar os resultados alcançados.

Palavras chaves: cartão de crédito, fraudes, *machine learning*, aprendizado de máquina, supervisionado, SVM, KNN.

Abstract

Each year, the use of credit cards as a form of payment and is generated as a transaction of criminals, who are in search of forms of fraud as transactions. New methods to combat these crimes are always available to reduce emissions from credit card providers. The big problem is that there are thousands of transactions per minute, some frauds go unnoticed. In this work, it will be necessary to study machine learning techniques that can be analyzed, as well as to identify possible frauds. Apply the techniques that remote the best result in the detection of fraud. Analyze and compare work related to the topic and show the results achieved.

Keywords: credit card, fraud, machine learning, machine learning, supervision, SVM, KNN.

Lista de Ilustrações

Lista de Tabelas

Lista de Siglas

IA	Inteligência Artificial
EUA	Estados Unidos da América
KNN	K-ésimo vizinho mais próximo
SVM	Máquina de vetores de suporte
USD	Dólar Americano
ABECS	Associação Brasileira das Empresas de Cartões de Crédito e Serviços
Qtd	Quantidade
PCA	Principal Component Analysis
CPF	Cadastro de Pessoa Física

Lista de Símbolos

V	Conjunto de Vetores
X	Conjunto de Variáveis

Sumário

1. Introdução

De acordo com os índices divulgados pelo Banco Central em julho de 2017, as transações com cartão de crédito somaram R$ 674 bilhões, o que representa um aumento de 3% em relação a 2015. Esse crescente número de transações com o uso do cartão de crédito resulta em alguns problemas que são as fraudes.

Segundo o Serasa, em 2017 ocorreram 1,96 milhão de tentativas de fraude, sendo o maior índice em três anos. O número representa ainda alta de 8,2% em relação a 2016. Dentre essas tentativas se encontra a modalidade com o cartão de crédito sendo emitido usando uma identificação falsa ou roubada. Cada cliente possui um padrão de compra. Encontrar esse padrão para cada cliente que as operadoras de cartão possuem e identificar uma transação suspeita para cada cliente é um desafio para elas. Com isso, gera prejuízos que ficam para o governo, comerciantes, organizações corporativas, instituições financeiras etc.

Dessa forma, a inteligência artificial pode contribuir para esse problema, visto que segundo (RICH & KNIGHT, 1992) é o estudo de como fazer os computadores realizarem coisas que, no momento, as pessoas fazem melhor. Em inteligência artificial há um subcampo que é *Machine Learning* que são técnicas de aprendizado, capazes de adquirir conhecimento automático.

Portanto, dentre as técnicas de *machine learning* escolhidas nessa pesquisa, qual delas pode-se identificar uma transação fraudulenta com uma boa acuracidade?

O objetivo geral deste trabalho é aplicar técnicas de aprendizado de máquina escolhida com objetivo na acurácia na identificação de transação de cartão de crédito fraudulenta. Mais especificamente este trabalho irá:

- Classificar tipos de fraudes de cartão de crédito;
- Identificar técnicas de aprendizado de máquina que melhor atendem ao problema;
- Realizar o treinamento da máquina com as técnicas de aprendizado de máquina escolhidas no problema;
- Realizar análise comparativa dos resultados das técnicas aplicadas.

O aprendizado de máquina está crescendo cada vez mais entre as empresas em diferentes áreas por promover ganhos significativos como eficiência, produtividade e

segurança. Com o aprendizado de máquina é possível analisar um grande volume de dados complexos e aprender com eles. Além disso, é possível criar algoritmos que possam detectar quando algo está fora do padrão com grande precisão e aumentando essa precisão cada vez mais com a prática. Nos últimos anos o aprendizado de máquina teve uma evolução significativa devido à grande demanda de dados e o poder de processamento dos computadores atuais.

Algoritmos baseados em aprendizado de máquina estão sendo usados para detecção de anomalias em diversas áreas. Uma solução para o problema de fraude seria o uso desses algoritmos. Desenvolver e aplicar estes algoritmos em uma base de dados para conseguir identificar se a transação é fraudulenta ou não pode diminuir o prejuízo diário que as empresas levam, tentando coibir cada vez mais essa prática de fraude.

O uso de cartões (crédito ou débito) como forma de pagamento online ou presencial vem crescendo entre consumidores que buscam comodidade e segurança. Com o aumento do uso cresce também o interesse em fraudar esse processo. Os criminosos estão sempre inovando nas formas de fraude.

Para compras online são precisos poucos dados para efetuar o pagamento com cartões, o que facilita para o criminoso que consegue acesso a essas informações na internet. Cabe ao varejista e às operadoras de cartões monitorar as compras e adotar novos métodos de prevenção de fraude. Um sistema de monitoramento das compras além de ajudar a evitar fraudes pode coibir a ação dos criminosos se for levado em conta que a maioria dos casos não são notificados à polícia. O proprietário do cartão entra em contato direto com a empresa de crédito e tenta resolver entre eles mesmos. Sem o registro do boletim de ocorrência, as autoridades não podem dar início às investigações.

Com a detecção das fraudes, a própria administradora do cartão pode contribuir com as autoridades nas investigações. Além de evitar as fraudes, as administradoras vão contribuir para diminuir o número de fraudadores.

2. Referencial Teórico

2.1. Cartão de Crédito

Segundo Araujo (2010?), o cartão de crédito é uma forma de empréstimo com prazo de pagamento de até 40 dias, e disponibilizado por bancos e instituições financeiras. Quando você pede o cartão, recebe um limite de crédito para fazer compras de bens e serviços.

Figura 1 - Primeira versão do cartão Diners em 1950

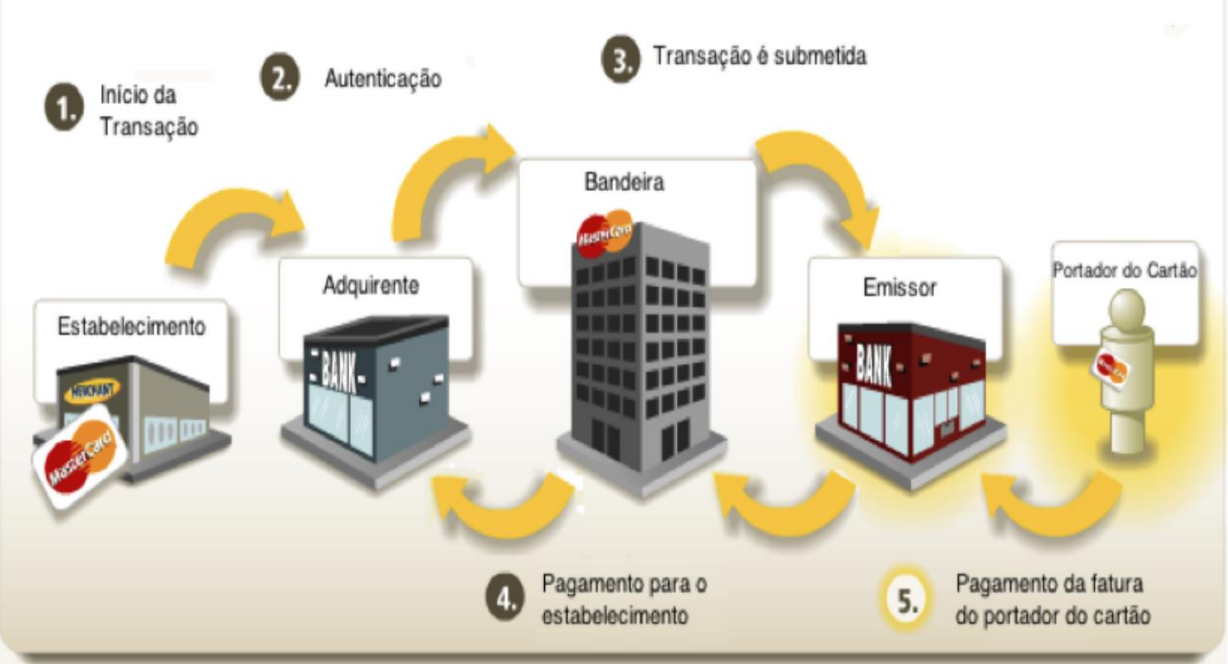
Fonte: http://www.fraudes.org/images/Diners_1950_card.jpg.

A Figura 1 apresenta a primeira versão em papel cartão do Diners Club Card. Segundo De Moraes (2008), essa versão trazia o nome do associado de um lado e dos estabelecimentos filiados no outro. Com este sistema a empresa de cartões de crédito cobrava uma taxa anual e enviava contas mensais ou anuais dos gastos efetuados.

2.2. Funcionamento do Cartão de Crédito

Para que o cliente consiga realizar uma compra com o cartão com sucesso, a transação passa por um processo de autorização pela operadora do cartão. Para isso, Gadi (2006) ressalta que quando um cliente utiliza seu cartão de crédito, no mesmo instante é transmitido um sinal para o adquirente que repassa este sinal para a bandeira, onde a emissora do cartão recebe esse sinal e por meio de critérios próprios decide-se por aprovar ou negar a transação.

Figura 2 - Passos para uma transação de cartão de crédito



Fonte: MasterCard.

A Figura 2 ilustra o processo que é executado para aprovar uma compra, que após a aprovação da compra, a cada mês é gerada uma fatura de consumo que o cliente realizou durante o mês para ser pago.

2.3. Fraudes com cartão de crédito

O cartão de crédito é uma das principais formas de pagamento em todo o mundo. Segundo a ABECS, em 2017 os brasileiros realizaram R$ 720 bilhões em compras com cartões no primeiro semestre de 2018, crescimento de 13,6% ante o mesmo período de 2017. Com o crescimento do uso dos cartões de crédito cresce o número de criminosos tentando ganhar vantagens sobre as transações. Existem inúmeras formas para fraudar o cartão. Entre elas temos clonagem de cartões e roubo de dados sigilosos. O que pode diferenciar uma fraude da outra é o método e como ela é realizada.

No roubo de informações pessoais um dos métodos mais comuns é uma ligação feita pelo fraudador se passando pela administradora do cartão pedindo para confirmar dados pessoais como CPF, número do cartão, código de segurança e senha. Hoje em dia esse método não é tão eficaz em grande parte dos consumidores, porém, pessoas mais leigas no assunto e principalmente idosos passam esses dados sem muito questionamento. Outra forma de conseguir esses dados é com o vazamento de dados de cadastros de clientes de grandes empresas. Muitos criminosos vendem esses dados pela internet.

A clonagem de cartão também merece uma atenção especial, devido ao grande número de esquemas utilizados por criminosos. Não há como definir um único método de clonagem, porém, os criminosos estão sempre tentando descobrir novos métodos, e as empresas sempre trabalhando para proteger os seus clientes. Um dos métodos mais conhecidos é o “chupa cabra” em que um dispositivo acoplado nos leitores de cartão de caixas eletrônicos clona os cartões, de modo que podem ser utilizados posteriormente. Outra forma de conseguir esses dados é roubá-los de sites de compra não confiáveis, que salvam dados do cartão, ou alguns sites que são cópias de lojas oficiais, e oferecem produtos com preços bem abaixo do mercado. Porém, em alguns casos, os criminosos procuram falhas para poder invadir os sites de compras e roubar os dados do cliente.

2.4. Inteligência artificial

No século XIX surge a inteligência artificial oficialmente, especificamente em 1956 em uma conferência de verão no colégio Dartmouth. Entre os participantes foi proposto “um estudo durante dois meses, por dez homens, sobre o tópico inteligência artificial” (McCarthy, 1995). Entre os participantes, alguns nomes se destacam como John McCarthy, Marvin Minsky, Allen Newell e Herbert Simon. Não há uma definição que diga o que é inteligência artificial, na verdade há uma série de práticas e técnicas que as pessoas juntam para definir a tecnologia.

Basicamente quando uma máquina desenvolve habilidades como aprender e raciocinar ela é considerada artificialmente inteligente. "Inteligência Artificial é o estudo de como fazer os computadores realizarem coisas que, no momento, as pessoas fazem melhor” (RICH & KNIGHT, 1992). Em outras palavras a inteligência artificial trata-se de uma inteligência não biológica.

Nos anos 50, Alan Turing criou o teste de Turing, pensando em como descobrir se uma máquina era inteligente ou não. A sua principal função era fazer com que a máquina se passasse por um humano em um interrogatório. Para que o computador pudesse passar no teste teria que persuadir o interrogador com alguns truques como errar respostas e se contradizer. O teste era feito com um interrogador em uma sala e na outra sala um computador e um humano e responderiam da mesma forma (datilografia). No final o interrogador deveria apontar quem era o computador. Caso não fosse possível, a máquina era considerada inteligente.

A inteligência artificial está em nosso dia a dia, nos dispositivos inteligentes que possuem tecnologia da internet das coisas, que para Deidmar (2017) é um conceito tecnológico em que todos os objetos da vida cotidiana estariam conectados à internet de modo inteligente e sensorial. Esta técnica está baseada em aprender o comportamento e o padrão de uso mostrado pelo usuário. Esta tecnologia consiste na ideia da fusão do "mundo real" com o "mundo digital", fazendo com que o indivíduo possa estar em constante comunicação e interação, seja com outras pessoas ou objetos.

2.5. Aprendizado de Máquina

Machine learning, ou aprendizado de máquina, para Monard (2000?, p. 1) é:

> Uma área de inteligência artificial cujo objetivo é o desenvolvimento de técnicas computacionais sobre o aprendizado bem como a construção de sistemas capazes de adquirir conhecimento de forma automática. Um sistema de aprendizado é um programa de computador que toma decisões baseado em experiencias acumuladas através da solução bem-sucedida de problemas anteriores. Os diversos sistemas de aprendizado de máquina possuem características particulares e comuns que possibilitam sua classificação quanto à linguagem de descrição, modo, paradigma e forma de aprendizado utilizada.

Então, basicamente aprendizado de máquina é uma área da inteligência artificial que busca estudos e desenvolvimentos de algoritmos que possam compreender de maneira autônoma. O aprendizado de máquina é capaz de entender e extrair padrões de um volume de dados que possivelmente seria de difícil compreensão para um humano ou levaria muito tempo. Após a compreensão desses dados a máquina será capaz de executar ações complexas com muito mais precisão e baseadas em uma lógica desenvolvida por ela. Além disso, quanto maior a prática da máquina maior será a sua precisão.

Enquanto na inteligência artificial existem os raciocínios indutivos e dedutivos. Em aprendizado de máquina o tipo de raciocínio é apenas o indutivo, que é a forma de inferência lógica que permite obter conclusões a partir de um conjunto de exemplos.

Segundo Monard (2000?), a inferência indutiva é um dos principais métodos utilizados para derivar conhecimento novo e predizer eventos futuros. A aprendizagem indutiva pode ser dividida em supervisionada e não supervisionada. No aprendizado supervisionado o programa é treinado sobre uma base de dados pré-definida, que baseado no treinamento com os dados pré-definidos, o programa pode tomar decisões precisas quando recebe novos dados. Já no aprendizado não supervisionado o programa pode automaticamente encontrar padrões e relações em um conjunto de dados.

Este trabalho irá tratar somente do aprendizado supervisionado, já que esse tipo de aprendizagem é dado em um conjunto de dados rotulados que já sabemos qual é a nossa saída correta e que deve ser semelhante ao conjunto, tendo a ideia de que existe uma relação entre a entrada e a saída.

2.5.1 Aprendizado supervisionado

Em aprendizado supervisionado, algoritmos são utilizados para induzir modelos preditivos por meio da observação de um conjunto de objetos rotulados (Von Luxburg e Schölkopf, 2008), tipicamente referenciado como conjunto de treinamento. Esses algoritmos podem ser divididos em “regressão” e “classificação”.

Em classificação o algoritmo vai criar um classificador capaz de aprender com informações já apresentadas no treinamento, com a finalidade de dizer se um objeto pertence ou não a uma determinada classe. Os e-mails utilizam um exemplo de classificação para determinar se um e-mail é ou não “spam”.

Quando um valor é contínuo, ou seja, algo esperado, o aprendizado supervisionado é chamado de regressão. Por exemplo, como acontece no preço de cotações, baseado em um conjunto de dados, o algoritmo pode prever o índice da inflação para um determinado período.

Neste estudo, foi examinado ecomparado os desempenhos dos classificadores KNN, SVM e Naive bayes para a detecção de fraudes com o uso de cartão de crédito em diferentes cenários de experimentos.

2.6. Máquina de vetores de suporte (SVM)

Para James (2013), o SVM é uma técnica de aprendizado supervisionado que analisa os dados e reconhece padrões, usado para a classificação e análise de regressão, mas é muito utilizado em classificação. O SVM é uma representação de exemplos com pontos no espaço, mapeados de maneira que os exemplos de cada categoria sejam divididos por um espaço claro que seja tão amplo quanto possível. Os novos exemplos são então mapeados no mesmo espaço e preditos como pertencentes a uma categoria baseados em qual lado do espaço eles são colocados.

2.6.1. Problema Linear

Um problema linear é quando ocorre na obtenção de fronteiras lineares para a separação de dados pertencentes a duas classes. Segundo Carvalho (2007), as SVMs

lineares com margens rígidas definem fronteiras lineares a partir de dados linearmente separáveis. Classificadores que separam os dados por meio de um hiperplano são denominados lineares.

2.6.2. Hiperplano

Para entender o que é hiperplano se faz necessário saber o que é espaço e subespaço vetorial. Espaço, segundo Pinto (2005), é uma entidade formada pelo conjunto dos números reais, um conjunto de vetores e uma operação entre esses conjuntos.

De uma forma mais simples, Espaço Vetorial é uma "coleção" (conjunto) de vetores que obedece a 8 axiomas, sendo 4 aditivos e 4 multiplicativos: Axiomas: Sendo V um conjunto de vetores e o conjunto dos números reais, dizemos que V é um Espaço Vetorial se os axiomas são obedecidos.

Já o subespaço vetorial, de acordo com Pinto (2005), são subconjuntos vetoriais que obedecem aos axiomas apresentados anteriormente. Mas o fato de serem subconjuntos nos permite apenas verificar quatro propriedades para que possamos provar (ou negar) que são Subespaços Vetoriais.

Segundo James (2013), um hiperplano é um subespaço afim plano do hiperplano dimensão em um espaço p-dimensional. Por exemplo, em duas dimensões, um hiperplano é um plano subespaço unidimensional - em outras palavras, uma linha. Em três dimensões, um hiperplano é um subespaço bidimensional plano - isto é, um plano. A definição matemática de um hiperplano é bastante simples. Segundo Rio (2015?), é definido pela equação 1:

$$w.x + b = 0 \tag{1}$$

onde $w \in \mathbb{R}^n$ é o vetor de pesos e o escalar b é o bias. Um vetor x_j da classe y_j, $y_j \in \{-1,1\}$, deve satisfazer a equação 2.

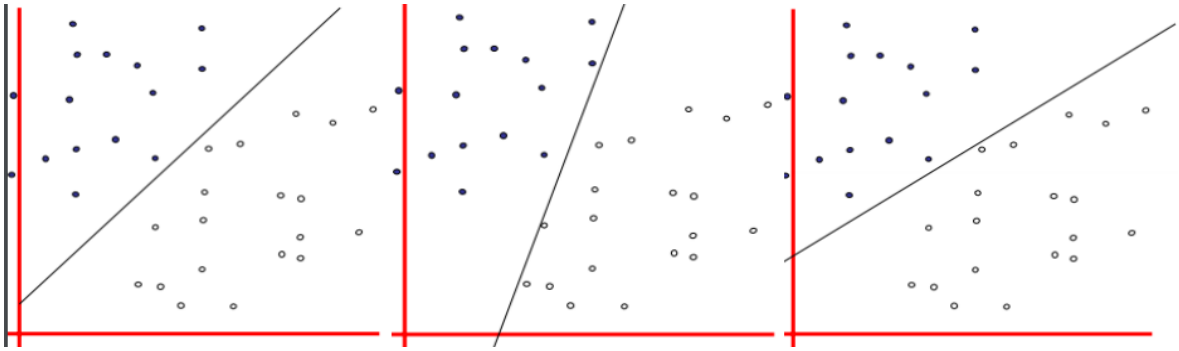
$$y_j((w.\mathrm{x}_j) + b) \geq para\ j = 1,\ldots,N \tag{2}$$

A distância euclidiana entre esse hiperplano e os pontos que estão sobre a margem, isto é, $y_j((w.x_j) + b) = 1$, é determinada pela equação 3:

$$\frac{y_j((w.\mathrm{x}_j) + b)}{||w||} = \frac{1}{||w||} \tag{3}$$

Assim, minimizar $||w||$ é equivalente a maximizar a margem do hiperplano de separação.

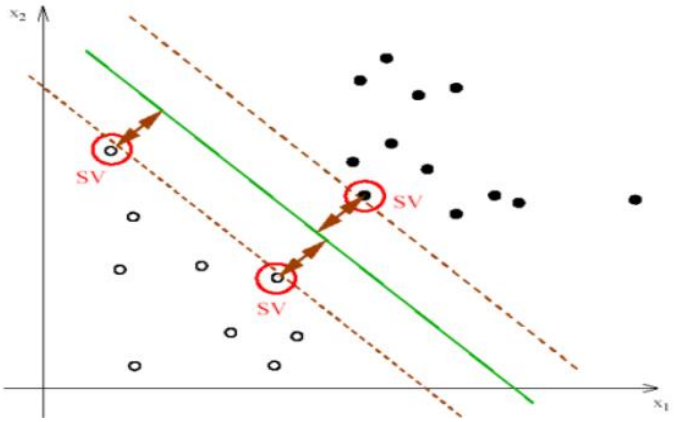
Figura 3 - Hiperplanos possíveis



Fonte: Machine Learning e Data Science com Python.

A Figura 3 mostra um mesmo conjunto de dados, mas sendo divididos pelas categorias com retas distintas sendo tracejadas em cada hiperplano. O que o SVM faz é encontrar a melhor reta possível, com a separação com margem máxima dos vetores de suporte, como pode-se ver na Figura 4.

Figura *4* - Vetores de Suporte



Fonte: Machine Learning e Data Science com Python.

Os vetores de suporte (SV) são os responsáveis por encontrar a melhor reta, então por meio de cálculos matemáticos são encontrados esses vetores, para que logo em seguida seja possível definir a reta de separação do hiperplano.

O SVM é basicamente um método que executa tarefas de classificação por meio da construção de hiperplanos em um espaço multidimensional que separa casos de rótulos de classes diferentes. O SVM oferece suporte a tarefas de regressão e classificação e pode manipular várias variáveis contínuas e categóricas. Para variáveis categóricas, uma variável fictícia é criada com valores de caso como 0 ou 1.

Conclui-se, segundo Carvalho (2007), que na geração de um classificador linear deve-se buscar um hiperplano que tenha margem máxima e cometa poucos erros marginais. Esse hiperplano é denominado ótimo.

4.7. *Naive Bayes*

O Algoritmo de Naive Bayes é um classificador probabilístico baseado no "Teorema de Bayes", o qual foi criado por Thomas Bayes (1701 – 1761) para tentar provar a existência de Deus.

A aprendizagem bayesiana, segundo Stuart (2013):

> simplesmente calcula a probabilidade de cada hipótese, considerando-se os dados, e faz previsões de acordo com ela. Isto é, as previsões são feitas com o uso de todas as hipóteses, ponderadas por suas probabilidades, em vez de utilizar apenas uma única "melhor hipótese". Dessa forma, a aprendizagem é reduzida à inferência probabilística.

Uma das características dessa aprendizagem é que ela desconsidera completamente a correlação entre as variáveis. Além disso, o naive bayes só precisa de um pequeno número de dados de teste para concluir classificações com uma boa precisão.

Os classificadores Naive Bayes podem manipular um número arbitrário de variáveis independentes, contínuas ou categóricas. Dado um conjunto de variáveis, $X = \{x_1,\ x_2,\ x\ldots,\ x_d\}$, queremos construir a probabilidade posterior para o evento C_j entre um conjunto de resultados possíveis $C = \{c_1,\ c_2,\ c\ldots,\ c_d\}$. Em uma linguagem mais familiar, X é os preditores e C é o conjunto de níveis categóricos presentes na variável dependente. Usando a regra de Bayes definida pela a equação 4:

$$p(C_j \mid x_1, x_2, \ldots, x_\partial) \propto p(x_1, x_2, \ldots, x_\partial \mid C_j)p(C_j) \tag{4}$$

Na Equação 4, onde $p(C_j \mid x_1, x_2, \ldots, x_\partial)$ é a probabilidade posterior de pertencer à classe, isto é, a probabilidade de X pertencer a C_j. Como Naive Bayes assume que as probabilidades condicionais das variáveis independentes são estatisticamente independentes, podemos decompor a probabilidade para um produto de termos a Fórmula 5:

$$p(C_j \mid X) \propto p\,(C_j) \prod_{k=1}^{\partial} p(x_k \mid C_j) \tag{5}$$

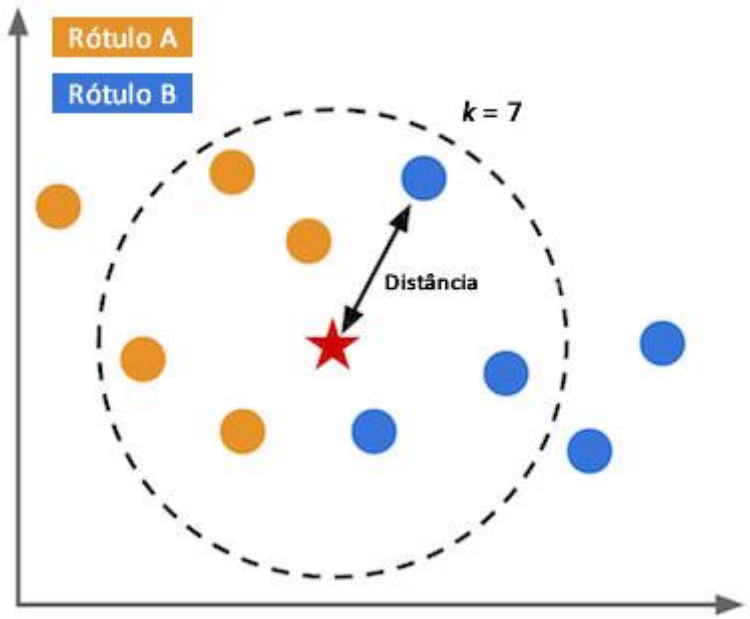
Usando a Equação 5 de Bayes, rotulamos um novo caso X com um nível de classe C_j que atinge a maior probabilidade posterior.

Embora a suposição de que as variáveis preditoras (independentes) sejam independentes nem sempre seja exata, ela simplifica a tarefa de classificação drasticamente, pois permite que as densidades condicionais de classe $p(x_k \mid C_j)$ sejam calculadas separadamente para cada variável, isto é, reduz uma tarefa multidimensional para um número de tarefas unidimensionais. Com efeito, Naive Bayes reduz uma tarefa de estimativa de alta densidade para uma estimativa de densidade de núcleo unidimensional. Além disso, a suposição não parece afetar grandemente as probabilidades posteriores, especialmente em regiões próximas a limites de decisão, deixando assim a tarefa de classificação inalterada.

2.8. K-ésimo Vizinho Mais Próximo (KNN)

Segundo Zhou (2009), o KNN é uma técnica usada para classificação e regressão, que tem como objetivo principal determinar o rótulo de classificação de uma amostra baseada nas amostras vizinhas advindas de um conjunto de treinamento.

Figura 5 - Exemplo de classificação do KNN com dois rótulos de classe e K = 7



Fonte: http://www.computacaointeligente.com.br/wp-content/uploads/2017/03/imgKnn.jpg.

Dois pontos chaves que devem ser determinados para aplicação do KNN são: a métrica de distância e o valor de K. Para métrica de distância a mais utilizada é a distância Euclidiana, descrita pela Fórmula 6:

$$D = \sqrt{(p_1 - q_1)^2 + \ldots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \quad (6)$$

Onde $p = (p_1, \ldots, p_n)$ e $p = (p_1, \ldots, q_n)$ são dois pontos *n* - dimensionais. No exemplo da Figura 1, essa distância seria calculada entre as bolinhas (azuis e laranjas) e a estrela (a nova entrada). Como o exemplo é 2D, um cada ponto teria seu valor em **x** e em **y**. Para problemas com dimensões maiores a abordagem é a exatamente a mesma.

O KNN é um modelo baseado em memória, definido por um conjunto de objetos conhecidos como exemplos (também conhecidos como instâncias) para os quais o resultado é conhecido (isto é, os exemplos são rotulados). Cada exemplo consiste em um caso de dados com um conjunto de valores independentes rotulados por um conjunto de resultados dependentes. As variáveis independentes e dependentes podem ser contínuas ou categóricas. Para variáveis dependentes contínuas, a tarefa é regressão; caso contrário, é uma classificação.

O KNN encontra exemplos que estão mais próximos da distância do ponto de consulta, portanto o nome K - vizinhos mais próximos. Para problemas de regressão, as previsões de KNN são baseadas na média dos resultados dos vizinhos K mais próximos. Para problemas de classificação, a maioria da votação é usada.

A escolha de K é essencial na construção do modelo KNN. De fato, K pode ser considerado como um dos fatores mais importantes do modelo que pode influenciar fortemente a qualidade das previsões. Para qualquer problema, um pequeno valor de K levará a uma grande variação nas previsões. Alternativamente, definir K para um valor maior pode levar a um grande viés do modelo. Assim, K deve ser configurado para um valor grande o suficiente para minimizar a probabilidade de erro de classificação e pequeno o suficiente (com relação ao número de casos na amostra de exemplo) para que os K pontos mais próximos estejam próximos o suficiente do ponto de consulta. Assim, como qualquer parâmetro de suavização, existe um valor ótimo para k que atinge o *trade-off* correto entre o viés e a variância do modelo. (Bishop, 2006).

2.9 PCA - *Principal Component Analysis*

A Principal Component Analysis, ou PCA, segundo Jaadi (2017?) é:

> um método de redução de dimensionalidade que é frequentemente usado para reduzir a dimensionalidade de grandes conjuntos de dados, transformando um grande conjunto de variáveis em um menor que ainda contém a maioria das informações no conjunto grande.

> Reduzir o número de variáveis de um conjunto de dados ocorre naturalmente à custa da precisão, mas o truque na redução de dimensionalidade é trocar um pouco de precisão pela simplicidade. Porque conjuntos de dados menores são mais fáceis de explorar e visualizar e tornar a análise de dados muito mais fácil e rápida para algoritmos de aprendizado de máquina sem que variáveis externas sejam processadas.

Então, com essa técnica pretende-se facilitar o aprendizado dos algoritmos.

3. Metodologia

Devido às necessidades e particularidades em cartões de crédito, técnicas de aprendizado de máquina já foram estudadas para o problema de detecção de fraudes em cartões. Oliveira (2016), em sua dissertação de fraude de cartão de crédito, utiliza as técnicas de regressão baseado em regras para sua avaliação, a análise da base de dados real e original selecionando as tabelas e campos para o interesse do trabalho com a ajuda de especialistas no domínio da empresa, aplicação de filtros e criação de variáveis derivadas para uma base final para a modelagem, sendo dividida em amostras de treinamento e amostras de teste.

Com a base de modelagem obtida, foi feito um processo de amostragem, que se faz necessário, pois é preciso reduzir o volume de dados para que o trabalho seja factível tanto em termos de recursos computacionais quanto temporais. Oliveira (2016) se deparou com o fenômeno de desbalanceamento de classes, que significa que há relativamente poucas transações fraudulentas para muitas transações legítimas.

O próximo passo foi a segmentação dos dados entre amostra de treinamento e amostra de teste de validação. A execução do algoritmo *FP-Growth*, como objetivo inicial, era encontrar regras de associação que levassem determinadas características das transações às fraudes. Com modificações na implementação do algoritmo *FP-Growth*, teve como resultado da aplicação desse algoritmo no conjunto de treinamento que somente com transações fraudulentas foram geradas mais de 38 mil regras, que a partir daí foi preciso disparar cada uma dessas regras no conjunto de treinamento contendo somente transações legítimas.

Com a grande quantidade de regras obtidas, foi suficiente para inviabilizar uma análise regra a regra, e logo foi preciso criar estratégias para diminuir o conjunto de regras. Com a mineração de regras de associação e cada um dos conjuntos de regras resultante do procedimento realizado foi submetido a modelagem com utilização de técnica de regressão logística.

Os conjuntos de regras minerados foram individualmente submetidos ao modelo de regressão logística, onde foi utilizado o *software SAS Enterprise Guide 5.1*. A modelagem de regressão logística consistiu nas seguintes atividades:

1. Preparar os dados;
2. Ajustar os modelos;
3. Definir os limiares de classificação;
4. Validar os modelos;
5. Avaliar e comparar os indicadores de desempenho.

Após a realização dessas etapas utilizando a técnica baseada em regras e regressão logística, foi demonstrada sua viabilidade técnica nas fases de modelagem e de avaliação, tendo como resultado predições bastante razoáveis chegando a obter uma porcentagem de acurácia de quase 80%.

Patil (2018) em seu trabalho de criação de um modelo de predição de fraude de cartão de crédito usou dois componentes para detecção de fraudes em tempo real, tendo um como responsabilidade pelo processamento dos dados e o servidor analítico para a modelagem preditiva. O sistema se baseia em uma rede Hadoop que armazena dados no HDFS provenientes de diferentes fontes. Os dados são lidos e convertidos para dados brutos que após é levado para um modelo analítico para a construção do modelo de dados. Isso torna o sistema altamente escalável e ajuda a construir um forte modelo analítico de autoaprendizagem em tempo real.

O modelo analítico verifica se a transação recebida é legitima ou não e os modelos de regressão logística e de aprendizado de máquina na árvore de decisão são implementados para a detecção de fraudes.

Foi usada a regressão logística para a detecção de fraude, pois é um tipo de modelo probabilístico de classificação estatística e utiliza a curva logística para a detecção de fraudes. Já a arvore de decisão usa a técnica de ID3 para construir considerando a entropia do conjunto de dados. A entropia é usada para medir a quantidade de incerteza no conjunto de dados.

Os modelos são executados no conjunto de dados de fraude de cartão de crédito e a precisão do modelo analítico é avaliada com a ajuda da matriz de confusão. A matriz de confusão informa como os registros nos modelos de treinamento e teste são classificadas corretamente. As predições são boas com a regressão logística chegando a quase 70% e a

precisão da árvore de decisão chegando a 72%. Se fossem considerados outros parâmetros estatísticos do modelo, o modelo de árvore de decisão seria melhor em comparação com o modelo de regressão logística porque o modelo de árvore de pontos de conjunto de dados em região menor de acordo com a divisão do atributo com base na entropia, enquanto a regressão logística se ajusta a uma única linha para dividir os pontos do conjunto de dados exatamente em duas regiões com base na probabilidade de limiar.

O que se pode ver nesses dois trabalhos foi o uso de duas técnicas específicas, a regressão logística e a arvore de decisão, que foram aplicadas de formas diferentes, sendo a primeira a geração de regras de decisão e após a aplicação da regressão logística para buscar as melhores regras com base no conjunto de dados. Já o segundo trabalhou os modelos que foram aplicados de forma individual, tendo a árvore de decisão gerando melhores resultados.

Serão utilizadas técnicas diferentes das que foram utilizadas nos trabalhos relatados em busca de melhores resultados para o problema e, dentre todos os métodos existentes, os métodos que iremos trabalhar são identificados como métodos de aprendizado conjunto. São identificados como métodos populares comuns, não por sua implementação bastante direta, mas também devido a seu excepcional desempenho preditivo em problemas práticos.

3.1. Métricas Utilizadas

Para calcular a métricas que serão apresentadas a seguir, foi utilizada a biblioteca *scikit-learn* que já possui a implementação das métricas que serão utilizadas para avaliar a performance dos algoritmos.

3.1.1 Tabela de Confusão

A tabela de confusão, segundo Santana (2018), é uma matriz de valores reais e valores preditos pelo classificador e nos permite realizar uma análise mais detalhada da situação do nosso classificador, uma vez que ela distingue nossos resultados em quatro classes.

- ***True positive* (TP) ou Verdadeiros Positivos (VP):** casos em que retornamos a classe FRAUDULENTA e a transação realmente era de fraudulenta.
- ***False positives* (FP) ou Falsos positivos (FP)**: casos em que retornamos a classe FRAUDULENTA e na verdade eram transações não fraudulentas.

- ***True Negative*** **(TN) ou Falsos Verdadeiros (FV):** casos que retornamos que era transação normal e realmente eram.
- ***False Negative*** **(FN) ou Falsos Negativos (FN):** retornamos que não eram fraudulentas e na verdade eram fraudulentas.

3.1.2 Acurácia (*accuracy*)

Leal (2017) define a acurácia como sendo o número de acertos dividido pelo número total de teste, ou seja, é a precisão e exatidão de dados e informações quando há ausência de erros ou equívocos. Obtemos a acurácia com a Fórmula 7:

$$Acurácia = \frac{Verdadeiros\ Positivos\ (VP) + Verdadeiros\ Negativos\ (VN)}{Total} \quad (7)$$

3.1.3 Precisão (*precision*)

Precisão é o grau de concordância entre resultados de medição obtidos com o mesmo procedimento. Leal (2017) diz que apesar de alguns traduzirem a métrica anterior (*accuracy*) como precisão, vamos utilizar apenas a descrição a seguir: daqueles que classifiquei como corretos, quantos efetivamente eram? Traduzindo em Fórmula 8:

$$Precisão = \frac{Verdadeiros\ Positivos\ (VP)}{Verdadeiros\ Positivos\ (VP) + Falsos\ Positivos\ (FP)} \quad (8)$$

3.1.4 *Recall*

Segundo Leal (2017), o recall é a frequência em que o seu classificador encontra os exemplos de uma classe, ou seja, "quando realmente é da classe X, o quão frequente você classifica como X?". Traduzindo para Fórmula 9:

$$Recall = \frac{Verdadeiros\ Positivos\ (VP)}{Verdadeiros\ Positivos\ (VP) + Falsos\ Negativos\ (FN)} \quad (9)$$

3.1.5 *F1 Score*

Para Leal (2017), essa métrica combina precisão e recall de modo a trazer um número único que indique a qualidade geral do seu modelo e trabalha bem até com conjuntos de dados que possuem classes desproporcionais.

A fórmula que define o F1 é a Fórmula 10 (quanto maior melhor o modelo):

$$\mathrm{F1} = \frac{2 * precisão * recall}{precisão + recall} \quad (10)$$

3.2 Base de Dados

Para criação do algoritmo é necessária uma base de dados para o treinamento de *machine learning*. Para isso foi encontrado uma base de dados criada em setembro de 2013 na Europa durante uma pesquisa que teve a colaboração da *Wordline and the machine Learning Group da Université Libre de Bruxelles* (Credit, 2018). A base possui 284807 transações que ocorreram por dois dias, onde foram encontradas 492 fraudes disponível pelo o link: https://www.kaggle.com/mlg-ulb/creditcardfraud.

Ela contém apenas variáveis numéricas de entrada que são o resultado de uma transformação do PCA. Infelizmente, devido a problemas de confidencialidade, não foi possível fornecer os recursos originais e mais informações básicas sobre os dados. Os recursos V1, V2, ... V28 são os principais componentes obtidos com o PCA, os únicos recursos que não foram transformados com o PCA são '*Time*' e '*Amount*'. Recurso 'Tempo' contém os segundos decorridos entre cada transação e a primeira transação no conjunto de dados. O recurso '*Amount*' é o Montante da transação, esse recurso pode ser usado para o aprendizado de custo-dependente, dependente de exemplo. Recurso '*Class*' é a variável de resposta e leva valor 1 no caso de fraude e 0 caso contrário.

Essa base foi escolhida por ter registros reais de transações de cartão de crédito e para ser utilizada como treinamento dos algoritmos para analisar a performance dos algoritmos.

4. Trabalhos Correlatos

Utilizando pesquisa bibliográfica, apresentamos no capítulo 2 os conceitos de Inteligência Artificial, com o foco em técnicas de *machine learning* para a aplicação no problema. O método de pesquisa será exploratório com o objetivo de aplicar técnicas de *machine learning* com aprendizagem supervisionada para a detecção de transação de cartão de crédito fraudulenta.

Decidiu-se aplicar as seguintes técnicas de aprendizado de máquina: Máquina de vetores de suporte (SVM), Naive Bayes e K-ésimo vizinho mais próximo (KNN) pelo tempo de realização dessa pesquisa e por serem técnicas bastantes populares com um ótimo desempenho preditivo em problemas de classificação.

Será explicado o funcionamento de cada técnica citada, treinamento da máquina, com o objetivo de alcançar a que se obtém a melhor acurácia, ou seja, qual delas oferece o melhor resultado de assertividade na identificação de transações fraudulentas e realizar análise comparativa dos resultados obtidos.

5. Experimentação

Como estudado no capítulo 2, o conceito de inteligência artificial, *machine learning* e seus algoritmos foram escolhidos para a realização dos experimentos de identificação de fraudes por meio da base de dados encontrada para a realização desse estudo.

Os tópicos a seguir apresentam os resultados dos experimentos realizados em cada algoritmo e o cenário de experimento. Todos os experimentos realizados foram utilizados 75% da base para o treinamento do algoritmo e 25% para teste, os algoritmos utilizados para os experimentos se encontra na seção de apêndice, sendo o apêndice A o algoritmo do KNN o apêndice B do algoritmo SVM e o apêndice C do algoritmo *naive bayes*, os experimentos houve apenas a mudança da base de dados com a exclusão de registros. A base de dados possui uma variável chamada "Class" que é a resposta da transação, sendo o valor 1 da classe de transações fraudulentas e o valor 0 da classe não fraudulenta.

Ao final serão discutidas as métricas relacionadas a classe 1 e a classe 0, devido a base de dados estar desbalanceada, sendo que 99,82% dos registros são da classe 0. Conforme os dados que serão apresentados, a acurácia foi alta em todos os experimentos realizados, ou seja, uma assertividade de quase 100%. Mas acurácia é uma métrica de assertividade da performance geral do algoritmo. Quando se observa a assertividade do algoritmo na predição da classe 1, que é a classe de uma transação fraudulenta, podemos notar as diferenças de performance dos algoritmos. É o que vamos ver agora.

5.1 Ferramentas Utilizadas

Para a realização dos experimentos que será apresentado, foi utilizado uma máquina com as seguintes configurações:

- Modelo: Dell Inspiron 5547
- Memória: 8Gb DDR3
- HD: 1Tb
- Processador: Intel Core i7 – 4ª. geração 2.0GHz Cache 128MB
- Placa de vídeo: AMD Radeon R7 M265

Os algoritmos foram escritos na linguagem python com a versão 3.7.3, utilizando a IDE spyder versão 3.3.1. Para a construção dos algoritmos também foi utilizado duas bibliotecas para a leitura da base de dados, treinamento dos algoritmos e coleta dos resultados e métricas que foram:

- **scikit-learn** é uma biblioteca de aprendizado de máquina para treinamentos, predição e coleta de métricas dos algoritmos.
- **pandas** que é uma biblioteca utilizada para a manipulação e análise de dados.

5.2 Experimento com a base completa

Para a realização desse experimento foi considerada a base completa sem nenhuma modificação.

A tabela 1 mostra as métricas de performance do algoritmo KNN. Podemos ver a métrica *F1-Score*, que é o indicador de qualidade do modelo, que é calculado de acordo com o *precision* e o *recall*. Sendo assim, o algoritmo obteve uma qualidade de assertividade de 3% para a classe 1.

Tabela 1 - Métricas - Algoritmo KNN

Classe	***Precision***	***Recall***	***F1-Score***	**Qtd. Registros**
0	1.00	1.00	1.000	71082
1	1.00	0.02	0.03	120

Fonte: Os Autores

A tabela 2 mostra que o algoritmo acertou todos os registros da classe 0 (registros não fraudulentos) e acertou apenas 2 registros de 120 da classe 1 (registros fraudulentos), ou seja, o algoritmo acertou apenas 1,7%, resultado muito ruim para o problema que esse estudo se propõe a resolver.

Tabela 2 - Matriz de confusão - Algoritmo KNN

	0	**1**
0	71082	0
1	118	2

Fonte: Os Autores
Acurácia: 99.83%

Com o algoritmo SVM houve uma melhora, com um indicador de qualidade de 48% para a classe 1, como mostra a tabela 3.

Tabela 3 - Métricas - Algoritmo SVM

Classe	***Precision***	***Recall***	***F1-Score***	**Qtd. Registros**
0	1.00	1.00	1.000	71082
1	0.73	0.36	0.48	120

Fonte: Os Autores

A tabela 4 mostra que o algoritmo acertou a maioria dos registros da classe 0 e acertou 43 registros de 120 da classe 1, ou seja, o algoritmo obteve apenas 35,83% de assertividade, resultado razoável para o problema que esse estudo se propõe a resolver.

Tabela 4 - Matriz de confusão - Algoritmo SVM

	0	**1**
0	71066	16
1	77	43

Fonte: Os Autores
Acurácia: 99.86%

No experimento com o algoritmo *naive bayes* tivemos o melhor desempenho, mesmo com um indicador de qualidade baixo de 24%, como é mostrado na tabela 5.

Tabela 5 - Métricas - Algoritmo Naive Bayes

Classe	***Precision***	***Recall***	***F1-Score***	**Qtd. Registros**
0	1.00	0.99	1.000	71082
1	0.17	0.63	0.24	120

Fonte: Os Autores

A tabela 6 mostra um acerto de 76 registros de 120, com uma assertividade de 63,33%, resultado bom para o problema.

Tabela 6 - Matriz de confusão - Algoritmo Naive Bayes

	0	**1**
0	71633	449
1	44	76

Fonte: Os Autores
Acurácia: 99.30%

5.3 Experimento com menos registros

Para a realização desse experimento foram excluídos 150.000 registros da classe 0 de forma pseudoaleatória, com a intenção de equilibrar um pouco o desbalanceamento.

O algoritmo KNN continuou com o mesmo resultado do experimento anterior, com um resultado ruim de indicador de qualidade de 3%, como é apresentado na tabela 7.

Tabela 7 - Métricas - Algoritmo KNN

Classe	***Precision***	***Recall***	***F1-Score***	**Qtd. Registros**
0	1.00	1.00	1.000	33584
1	1.00	0.02	0.03	118

Fonte: Os Autores

Como é mostrado na tabela 8, o algoritmo acertou apenas 2 registros de 118, assertividade de 1,69%.

Tabela 8 - Matriz de confusão - Algoritmo KNN

	0	**1**
0	33584	0
1	116	2

Fonte: Os Autores

Acurácia: 99.65%

O SVM teve um decréscimo em sua performance em relação ao experimento anterior, atingindo um indicador de qualidade de 43%, como é mostrado na tabela 9.

Tabela 9 - Métricas - Algoritmo SVM

Classe	***Precision***	***Recall***	***F1-Score***	**Qtd. Registros**
0	1.00	1.00	1.000	33584
1	0.70	0.31	0.43	118

Fonte: Os Autores

Na tabela 10 é mostrado um acerto de 37 registros de 118, diminuindo para 31,35% de assertividade.

Tabela 10 - Métricas - Algoritmo SVM

	0	**1**
0	33568	16
1	81	37

Fonte: Os Autores

Acurácia: 99.71%

O *naive bayes* manteve a sua boa performance com o aumento do indicador de qualidade em relação ao experimento anterior para 33%, de acordo com a tabela 11.

Tabela 11 - Métricas - Algoritmo Naive Bayes

Classe	***Precision***	***Recall***	***F1-Score***	**Qtd. Registros**
0	1.00	0.99	1.000	33584
1	0.22	0.64	0.33	118

Fonte: Os Autores

Na tabela 12 o algoritmo acertou 76 registros de 118, atingindo 64,40% de assertividade.

Tabela 12 - Matriz de confusão - Algoritmo Naive Bayes

	0	**1**
0	33317	267
1	42	76

Fonte: Os Autores
Acurácia: 99.08%

5.4 Experimento com menos Colunas

Para a realização desse experimento foram excluídas 15 colunas/variáveis de forma pseudoaleatória, com a intenção de ajudar os algoritmos de possíveis colunas que podem prejudicar o algoritmo.

O KNN continuou com a sua má performance com o indicador de qualidade de 2%, como é mostrado na tabela 13.

Tabela 13 - Métricas - Algoritmo KNN

Classe	***Precision***	***Recall***	***F1-Score***	**Qtd. Registros**
0	1.00	1.00	1.000	71082
1	1.00	0.01	0.02	120

Fonte: Os Autores

O algoritmo acertou apenas um registro de 120, como é mostrado na tabela 14, gerando assim uma assertividade de 0,83%.

Tabela 14 - Matriz de confusão – Algoritmo KNN

	0	**1**
0	71082	0
1	119	1

Fonte: Os Autores
Acurácia: 99.83%

O SVM teve boa performance com o aumento do indicador de qualidade em relação ao experimento anterior para 22% de acordo com a tabela 15.

Tabela 15 - Métricas - Algoritmo SVM

Classe	*Precision*	*Recall*	*F1-Score*	Qtd. Registros
0	1.00	1.00	1.000	71082
1	0.37	0.16	0.22	120

Fonte: Os Autores

O indicador de qualidade do SVM caiu para 22% em relação aos experimentos anteriores, consequentemente diminuiu sua assertividade registrando 15,83%, acertando 19 registros de 120, como é mostrado na tabela 16.

Tabela 16 - Matriz de Confusão - Algoritmo SVM

	0	**1**
0	71049	33
1	101	19

Fonte: Os Autores
Acurácia: 99.85%

Tabela 17 - Métricas - Algoritmo Naive Bayes

Classe	*Precision*	*Recall*	*F1-Score*	Qtd. Registros
0	1.00	1.00	1.000	71082
1	0.29	0.47	0.36	120

Fonte: Os Autores

Houve um aumento no indicador de qualidade do algoritmo *naive bayes* registrando 36%, porém sua performance caiu registrando 46,66%, acertando 56 registros de 120, conforme a tabela 18.

Tabela 18 - Matriz de confusão - Algoritmo Naive Bayes

	0	**1**
0	71633	139
1	64	56

Fonte: Os Autores
Acurácia: 99.71%

5.5 Experimento com PCA

Para a realização desse experimento foi utilizado o algoritmo PCA, ou seja, a realização da otimização de variáveis. Nesse caso, de 30 variáveis e/ou colunas diminuindo

para cinco colunas. Isso com a intenção de enriquecer as variáveis de forma com que os algoritmos tenham boa performance.

O KNN continuou com a sua má performance, com o indicador de qualidade de 0,00%, como é mostrado na tabela 19 com o algoritmo aplicado no apêndice D.

Tabela 19 - Métricas - Algoritmo KNN

Classe	***Precision***	***Recall***	***F1-Score***	**Qtd. Registros**
0	1.00	1.00	1.000	71082
1	0.0	0.00	0.00	120

Fonte: Os Autores

E como é mostrado na tabela 20, o algoritmo não acertou nenhum registro de 120.

Tabela 20 - Matriz de confusão - Algoritmo KNN

	0	**1**
0	71082	0
1	120	0

Fonte: Os Autores

Acurácia: 99.83%

O SVM obteve um indicador de qualidade de 28%, como é mostrado na tabela 21 com o algoritmo aplicado no apêndice E.

Tabela 21 - Métricas - Algoritmo SVM

Classe	***Precision***	***Recall***	***F1-Score***	**Qtd. Registros**
0	1.00	1.00	1.000	71082
1	0.83	0.17	0.28	120

Fonte: Os Autores

Porém atingiu uma assertividade de 16,66%, acertando 20 registros de 120, como é mostrado na tabela 22 com o algoritmo aplicado no apêndice F.

Tabela 22 - Matriz de confusão - Algoritmo SVM

	0	**1**
0	71078	4
1	100	20

Fonte: Os Autores

Acurácia: 99.85%

O indicador de qualidade do *naive bayes* diminuiu para 13%, como mostra a tabela 23.

Tabela 23 - Métricas - Algoritmo Naive Bayes

Classe	***Precision***	***Recall***	***F1-Score***	**Qtd. Registros**
0	1.00	1.00	1.000	71082
1	0.08	0.36	0.13	120

Fonte: Os Autores

Porém atingiu uma assertividade baixa, registrando 35,83%, acertando 43 registros de 120, conforme a tabela 24.

Tabela 24 - Matriz de confusão - Algoritmo Naive Bayes

	0	**1**
0	70605	477
1	77	43

Fonte: Os Autores

Acurácia: 99.221%

6. Análise de Resultados

De acordo com os resultados dos experimentos do tópico anterior, em todos o *naive bayes* se sobressaiu. Ele teve uma adaptação melhor na base de dados, enquanto seus concorrentes não conseguiram obter bons resultados.

O problema da má performance dos algoritmos foi devido ao desbalanceamento da base de dados, porque os algoritmos, em geral, têm a tendência de produzir modelos de classificação que favorecem as classes com maior probabilidade de ocorrência, resultando em baixas taxas de reconhecimento para os grupos minoritários segundo (Castro e Braga, 2011).

O fato de os modelos apresentarem altas taxas de acurácia geral tendem a prejudicar os exemplos das classes minoritárias. (Castro e Braga, 2011) dizem que, na maioria das aplicações reais, detectar eventos anormais (ou interessantes) em uma população contendo grande número de eventos comuns é o principal objetivo. Tais aplicações comumente apresentam conjuntos de dados altamente complexos, como é o caso desse trabalho, onde as transações são majoritariamente não fraudulentas.

O KNN teve a pior performance dentre seus concorrentes, isso é, devido ao seu funcionamento que com base nas classes de seus vizinhos serem majoritariamente da classe 0 fez com que a predição seja contaminada e consequentemente não conseguiu se adaptar com a classe 1. Miao et al. (2014) diz que quando o conjunto de treino está

desequilibrado, os k vizinhos mais próximos da amostra de teste podem ter uma maioria do conjunto maior, resultando no caso de erro de classificação.

O SVM foi um pouco melhor que o KNN, com seu funcionamento de traçar uma reta de separação das classes e produzir uma saída binária. A função objetiva do algoritmo não conseguiu separar as classes de forma a conseguir uma boa predição, pois as informações não tem um padrão. Cho (2006) diz que quando o valor da função objetivo é convertido no valor binário é importante as informações sobre o padrão não serem perdidas, com a distância entre o padrão e a classe fronteira. Quanto o maior valor absoluto, mais padrão terá com a classe fronteira. Portanto, na agregação do valor da função, a saída do conjunto é determinada pela adição de todos os valores da função objetivo do indivíduo.

Naive bayes foi o algoritmo que teve os melhores resultados, atingindo uma assertividade de 63,33% com a base completa. Isso se deve por ele ser um algoritmo probabilístico e baseado na evidência das variáveis nos dados segundo *Machine* (2000). O classificador bayesiano é bastante insensível à estratificação, ou seja, uma independência entre as variáveis no modelo que uma variável não tem nenhuma relação com as outras.

7. Considerações Finais

Conforme apresentado, quem possui cartão de crédito está vulnerável a ser lesado por um algum tipo de fraude e os principais prejudicados são as organizações corporativas e instituições financeiras, pois são elas que fornecem esse meio de pagamento.

Tentar identificar e coibir essas fraudes se tornou um desafio para as empresas, com isso a tecnologia pode contribuir nessa identificação de fraudes por meio da inteligência artificial, onde uma máquina simula o pensamento humano e desenvolve habilidades de aprendizagem e raciocínio. Existem vários algoritmos de aprendizado de máquina e esse trabalho teve como enfoque os algoritmos KNN, Naive Bayes e o SVM. A escolha foi devido ao tempo de realização dessa pesquisa e por serem técnicas bastantes populares com um ótimo desempenho preditivo em problemas de classificação, como nesse caso.

Dessa forma, esse trabalho teve como objetivo aplicar os algoritmos escolhidos e analisar os resultados obtidos. Objetivo realizado com sucesso e conforme os dados apontados conclui-se nesse trabalho que, ao aplicar os três algoritmos, o naive bayes se sobressaiu.

O *Naive Bayes* obteve os melhores resultados nos experimentos realizados com diferentes ambientes, com o objetivo de buscar os melhores resultados de forma a extrair a melhor performance de cada algoritmo.

Teve o desafio de aprender de forma teórica o funcionamento de cada algoritmo. Após aplicar cada técnica, foi utilizada a linguagem *python* para o treinamento, criado um algoritmo para cada técnica, onde o tempo de execução dos algoritmos do KNN e do *naive bayes* foi pequeno, cerca de 1 minuto para realizar o treinamento e executar os testes de predição. O tempo de execução do treinamento do SVM foi maior, cerca de 20 minutos para executar os experimentos.

Portanto, devido ao tempo não foi possível aplicar outros cenários de experimentos, nem testes com outros algoritmos de aprendizado de máquina. Dessa forma, como sugestão de novas pesquisas pode-se aplicar mais cenário de testes com outros algoritmos, como, por exemplo, a técnica de *kernel* no algoritmo SVM e aplicar pré-processamento na base de dados para balancear as classes da base de dados.

Para futuros sistemas de detecção de fraude como deste estudo, um desafio será a obtenção da base de dados. Para tal é aconselhável uma parceria com uma instituição de cartão de crédito com o intuito de coletar dados de transações, após aplicar as técnicas de

aprendizado de máquina, de preferência o algoritmo naive bayes, pois foi quem apresentou melhores resultados.

8. Referências Bibliográficas

ARAUJO, Fernanda. **Cartão de Crédito: o que é e como funciona**. [*S. l.*], 2010?. Disponível em: https://www.serasaconsumidor.com.br/ensina/seu-credito/cartao-de-credito-o-que-e-e-como-funciona/. Acesso em: 13 nov. 2018.

BISHOP. Christopher M. **Pattern Recognition and Machine Learning**. Germany: Springer, 2006. 758 p. Disponível em: http://users.isr.ist.utl.pt/~wurmd/Livros/school/Bishop%20-%20Pattern%20Recognition%20And%20Machine%20Learning%20-%20Springer%20%202006.pdf>. Acesso em: 06 nov. 2018.

Castro, C.; Braga, A. **Aprendizado supervisionado com conjuntos de dados desbalanceados**. 2011. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0103-17592011000500002&lng=en&nrm=iso&tlng=pt>. Acesso em: 06 mar. 2019.

CHO, Pilsung Kang and Sungzoon. **EUS SVMs: Ensemble of Under-Sampled SVMs for Data Imbalance Problems**. Computer Science Series as the Proceedings of International Conference on Neural Information Processing, Coreia do Sul, p. 151-744, 20 jul. 2006. Disponível em: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.98.5097&rep=rep1&type=pdf. Acesso em: 2 maio 2019.

CREDIT Card Fraud Detection. [S. l.], 22 mar. 2018. Disponível em: https://www.kaggle.com/mlg-ulb/creditcardfraud. Acesso em: 24 out. 2018.

Deidmar, G., Sobreira, D. and Lima, W. (2018). **Internet das coisas na Educação**. 2017. Disponível em: <http://revista.faculdadeprojecao.edu.br/index.php/Projecao4/article/download/1007/840> Acesso em: 21 abr. 2018.

DE MORAES, Dalila. **Modelagem de Fraude em Cartão de Crédito**. 2008. 136 p. Dissertação (Mestrado ao Departamento de Estatística) - Universidade Federal de São

Carlos, São Carlos, 2008. Disponível em: <https://repositorio.ufscar.br/bitstream/handle/ufscar/4557/4329.pdf?sequence=1&isAllowed=y>. Acesso em: 17 dez. 2018.

GADI, Manoel Fernando Alonso, LAGO, Alair Pereira. **Tópicos em ciência da computação na detecção de fraude**. 2006

HAYKIN, Simon. **Redes Neurais: Princípios e Prática**. 2ª. ed. São Paulo: Bookman, 2007. 900 p.

HENRIQUE M. A. OLIVEIRA, Paulo. **Detecção de fraudes em cartões: um classificador baseado em regras de associação e regressão logística**. 2016. 117 f. Dissertação (Pós-Graduação em Ciência da Computação) - Universidade de São Paulo, São Paulo, 2016. Disponível em: <http://www.teses.usp.br/teses/disponiveis/45/45134/tde-01022016-204144/publico/Paulo_Oliveira_Mestrado_PPGCC.pdf>. Acesso em: 23 out. 2018.

HORST, Paulo Sergio. **Avaliação do conhecimento adquirido por algoritmos de aprendizado de máquina**. 1999. 104 p. Dissertação (Mestrado na área de ciências de computação e matemática computacional) - Instituto de ciências matemáticas e de computação - USP, São Carlos, 1999.

JAADI, Zakaria. **A step by step explanation of Principal Component Analysis**. [S. l.], 28 fev. 2017?. Disponível em: https://towardsdatascience.com/a-step-by-step-explanation-of-principal-component-analysis-b836fb9c97e2. Acesso em: 10 abr. 2019.

JAMES, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: With Applications in R*. Corrected edition. New York: Springer, 2013. Print.

JACSON, R.H.F & JONES A.W.T.: **"An Architecture for Decision Making in the Factory of the Future"**, INTERFACES 17, n.6, pp.15-28, 1987.

JUSTO, Gabriela. **Cartão de crédito, fraude e responsabilidade objetiva**. 2015. Disponível em: <https://gabrielajusto.jusbrasil.com.br/artigos/189455197/cartao-de-credito-fraude-e-responsabilidade-objetiva>. Acesso em: 18 out. 2018.

LEAL, Renato Dos Santos. **Métricas comuns em Machine Learning: como analisar a qualidade de chat bots inteligentes**. [S. l.], 8 maio 2017. Disponível em: https://medium.com/as-m%C3%A1quinas-que-pensam/m%C3%A9tricas-comuns-em-machine-learning-como-analisar-a-qualidade-de-chat-bots-inteligentes-introdu-57ff30424192. Acesso em: 7 abr. 2019.

McCarthy, J.(1995). **A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence**. 1995. Disponível em: <http://www-formal.stanford.edu/jmc/history/dartmouth/dartmouth.html>
Acesso em: 18 dez. 2018.

KURZWELL, R. ***The age of intelligent machines***, Cambridge, MA; The MIT Press, 1990.

MACHINE Learning from Imbalanced Data Sets 101. Nova York, 22 jun. 2000. Disponível em: https://www.aaai.org/Papers/Workshops/2000/WS-00-05/WS00-05-001.pdf. Acesso em: 2 maio 2019.

MENDES, Raquel Dias. **INTELIGÊNCIA ARTIFICIAL**: SISTEMAS ESPECIALISTAS NO GERENCIAMENTO DA INFORMAÇÃO. 1997, vol.26, n.1, pp. Disponível em: <http://dx.doi.org/10.1590/S0100-19651997000100006> Acesso em: 18 jun. 2018.

MONARD, Maria Carolina; BARANAUSKAS, José Augusto. **Conceitos sobre Aprendizado de Máquina**. (2000?) Disponível em: <http://dcm.ffclrp.usp.br/~augusto/publications/2003-sistemas-inteligentes-cap4.pdf>.
Acesso em: 21 abr. 2018.

MIAO, Zongxia *et al*. **An Improved KNN Algorithm for Imbalanced Data Based on Local Mean. Journal of Computational Information Systems 10**, [*S. l.*], p. 5139–5146,

10 dez. 2014. Disponível em: https://pdfs.semanticscholar.org/fc55/e517baf145a07ea939899da479c0db45564a.pdf. Acesso em: 2 maio 2019.

PATIL, S.; NEMADE, V.; SONI, P. **Predictive Modelling For Credit Card Fraud Detection Using Data Analytics**. 2017, Disponível em: <https://www.sciencedirect.com/science/article/pii/S1877050918309347>. Acesso em: 8 nov. 2018.

PINTO, Paulo. **Resumo das Aulas Teóricas de Álgebra Linear**. Instituto Superior Técnico, [S. l.], p. 19-22, 17 maio 2005. Disponível em: https://www.math.tecnico.ulisboa.pt/~ppinto/al0405/Alteoricasem.pdf. Acesso em: 20 maio 2019.

RICH, E; KNIGHT, K **Inteligência Artificial**. 2. ed. São Paulo, Berlim: Springer-Verlag, 1992.

REZENDE, Solange Oliveira (org). **Sistemas Inteligentes**: fundamentos e aplicações. São Paulo: Manole, 2005.

RIO, PUC. **Máquinas de Vetores Suporte**. Disponível em: <http://www2.dbd.puc-rio.br/pergamum/tesesabertas/0124872_06_cap_02.pdf>. Acesso em: 15 dez. 2015?.

STUART, R.; NORVIG, P. **INTELIGENCIA ARTIFICIAL**. Tradução. 3. ed. Rio de Janeiro: Elsevier, 2013.

SANTANA, Rodrigo. **Café com Código #20: Matriz de Confusão, Você Sabe Utilizar?**. [S. l.], 16 jan. 2018. Disponível em: http://minerandodados.com.br/index.php/2018/01/16/matriz-de-confusao/. Acesso em: 7 abr. 2019.

SANT´ANA, R. C. G.; SANTOS, P. L. V. A. C. **Transferência de informação**: análise de fatores para identificação do valor de unidades de conhecimento registrado. In: VIDOTTI, S. A. B. G. Tecnologia e conteúdos informacionais. São Paulo: Polis, 2004. p. 53-76.

SELLITTO, Miguel Afonso. *INTELIGÊNCIA ARTIFICIAL*: UMA APLICAÇÃO EM UMA INDÚSTRIA DE PROCESSO CONTÍNUO. 2002. 14 p. Artigo (Centro de Ciências Exatas e Tecnológicas) - Universidade do Vale do Rio dos Sinos, Unisinos, São Leopoldo, RS, 2017. Disponível em: <http://www.scielo.br/pdf/%0D/gp/v9n3/14574.pdf>. Acesso em: 18 jun. 2018.

VASCONCELLOS, Paulo. **O que é Machine Learning e como aprender sem gastar nada**. [2000?]. Disponível em: <https://paulovasconcellos.com.br/o-que-e-machine-learning-e-como-aprender-sem-gastar-nada-2e612f13102b>. Acesso em: 09 maio 2018.

VILELA, Fernanda. **Inteligência Artificial: Os desafios para tornar as máquinas cada vez mais humanas**. 2013. Disponível em: <https://agenciacienciaweb.wordpress.com/2013/04/12/inteligencia-artificial-os-desafios-para-tornar-as-maquinas-cada-vez-mais-humanas/>. Acesso em: 07 mar. 2018.

ZHOU Yong; SHIXIONG, Li Youwen and Xia, **An Improved KNN Text Classification Algorithm based on Clustering**, Journal of Computers, Vol. 4, No. 3, Março 2009.

Apêndices

Apêndice A – Código fonte do algoritmo KNN

```
# Importação da biblioteca pandas, para a leitura do arquivo CSV.
import pandas as pd

# Leitura da base de dados e atribuindo para a variável "Base"
base = pd.read_csv('creditcard.csv')

# Atribui para a variavel "previsores" as colunas/variáveis da
aplicação que no caso são 30 colunas.
previsores = base.iloc[:, 0:30].values

# Atribui para a variável "classe" a coluna/variável o rótulo dos
registros.
classe = base.iloc[:, 30].values

# Importação da função "train_test_split" que irá dividir a base de
dados em uma base de treinamento e outra para teste.
from sklearn.model_selection import train_test_split
previsores_treinamento, previsores_teste, classe_treinamento,
classe_teste = train_test_split(previsores, classe, test_size=0.25,
random_state=0)

# Importação e treinamento da máquina.
from sklearn.neighbors import KNeighborsClassifier
classificador = KNeighborsClassifier(n_neighbors=30,
metric='minkowski', p=10)
classificador.fit(previsores_treinamento, classe_treinamento)

# Teste da máquina, realização das predições.
previsoes = classificador.predict(previsores_teste)

# Importação de bibliotecas para extrair métricas de performance do
algoritmo.
from sklearn.metrics import confusion_matrix, accuracy_score,
roc_auc_score, classification_report
precisao = accuracy_score(classe_teste, previsoes)
matriz = confusion_matrix(classe_teste, previsoes)

# Mostra na tela as métricas e resultados do algoritmo.
print(classification_report(classe_teste, previsoes))
print(matriz)
```

Apêndice B – Código fonte do algoritmo SVM

```
# Importação da biblioteca pandas, para a leitura do arquivo CSV.
import pandas as pd

# Leitura da base de dados e atribuindo para a variável "Base"
base = pd.read_csv('creditcard.csv')

# Atribui para a variavel "previsores" as colunas/variáveis da
aplicação que no caso são 30 colunas.
previsores = base.iloc[:, 0:30].values

# Atribui para a variável "classe" a coluna/variável o rótulo dos
registros.
classe = base.iloc[:, 30].values

# Importação da função "train_test_split" que irá dividir a base de
dados em uma base de treinamento e outra para teste.
from sklearn.model_selection import train_test_split
previsores_treinamento, previsores_teste, classe_treinamento,
classe_teste = train_test_split(previsores, classe, test_size=0.25,
random_state=0)

# Importação e treinamento da máquina.
from sklearn.svm import SVC
classificador =  SVC(kernel = 'linear', random_state = 1)
classificador.fit(previsores_treinamento, classe_treinamento)

# Teste da máquina, realização das predições.
previsoes = classificador.predict(previsores_teste)

# Importação de bibliotecas para extrair métricas de performance do
algoritmo.
from sklearn.metrics import confusion_matrix, accuracy_score,
roc_auc_score, classification_report
precisao = accuracy_score(classe_teste, previsoes)
matriz = confusion_matrix(classe_teste, previsoes)

# Mostra na tela as métricas e resultados do algoritmo.
print(classification_report(classe_teste, previsoes))
print(matriz)
```

Apêndice C – Código fonte do algoritmo *Naive Bayes*

```
# Importação da biblioteca pandas, para a leitura do arquivo CSV.
import pandas as pd

# Leitura da base de dados e atribuindo para a variável "Base"
base = pd.read_csv('creditcard.csv')

# Atribui para a variavel "previsores" as colunas/variáveis da
aplicação que no caso são 30 colunas.
previsores = base.iloc[:, 0:30].values

# Atribui para a variável "classe" a coluna/variável o rótulo dos
registros.
classe = base.iloc[:, 30].values

# Importação da função "train_test_split" que irá dividir a base de
dados em uma base de treinamento e outra para teste.
from sklearn.model_selection import train_test_split
previsores_treinamento, previsores_teste, classe_treinamento,
classe_teste = train_test_split(previsores, classe, test_size=0.25,
random_state=0)

# Importação e treinamento da máquina.
from sklearn.naive_bayes import GaussianNB
classificador = GaussianNB()
classificador.fit(previsores_treinamento, classe_treinamento)

# Teste da máquina, realização das predições.
previsoes = classificador.predict(previsores_teste)

# Importação de bibliotecas para extrair métricas de performance do
algoritmo.
from sklearn.metrics import confusion_matrix, accuracy_score,
roc_auc_score, classification_report
precisao = accuracy_score(classe_teste, previsoes)
matriz = confusion_matrix(classe_teste, previsoes)

# Mostra na tela as métricas e resultados do algoritmo.
print(classification_report(classe_teste, previsoes))
print(matriz)
```

Apêndice D – Código fonte do algoritmo KNN com PCA

```
# Importação da biblioteca pandas, para a leitura do arquivo CSV.
import pandas as pd

# Leitura da base de dados e atribuindo para a variável "Base"
base = pd.read_csv('creditcard.csv')

# Atribui para a variavel "previsores" as colunas/variáveis da
aplicação que no caso são 30 colunas.
previsores = base.iloc[:, 0:30].values

# Atribui para a variável "classe" a coluna/variável o rótulo dos
registros.
classe = base.iloc[:, 30].values

# Aplicação do PCA
from sklearn.decomposition import PCA
pca = PCA(n_components=5)
previsores = pca.fit_transform(previsores)

# Importação da função "train_test_split" que irá dividir a base de
dados em uma base de treinamento e outra para teste.
from sklearn.model_selection import train_test_split
previsores_treinamento, previsores_teste, classe_treinamento,
classe_teste = train_test_split(previsores, classe, test_size=0.25,
random_state=0)

# Importação e treinamento da máquina.
from sklearn.neighbors import KNeighborsClassifier
classificador = KNeighborsClassifier(n_neighbors=30,
metric='minkowski', p=10)
classificador.fit(previsores_treinamento, classe_treinamento)

# Teste da máquina, realização das predições.
previsoes = classificador.predict(previsores_teste)

# Importação de bibliotecas para extrair métricas de performance do
algoritmo.
from sklearn.metrics import confusion_matrix, accuracy_score,
roc_auc_score, classification_report
precisao = accuracy_score(classe_teste, previsoes)
matriz = confusion_matrix(classe_teste, previsoes)

# Mostra na tela as métricas e resultados do algoritmo.
print(classification_report(classe_teste, previsoes))
print(matriz)
```

Apêndice E – Código fonte do algoritmo SVM com PCA

```
# Importação da biblioteca pandas, para a leitura do arquivo CSV.
import pandas as pd

# Leitura da base de dados e atribuindo para a variável "Base"
base = pd.read_csv('creditcard.csv')

# Atribui para a variavel "previsores" as colunas/variáveis da
aplicação que no caso são 30 colunas.
previsores = base.iloc[:, 0:30].values

# Atribui para a variável "classe" a coluna/variável o rótulo dos
registros.
classe = base.iloc[:, 30].values

# Aplicação do PCA
from sklearn.decomposition import PCA
pca = PCA(n_components=5)
previsores = pca.fit_transform(previsores)

# Importação da função "train_test_split" que irá dividir a base de
dados em uma base de treinamento e outra para teste.
from sklearn.model_selection import train_test_split
previsores_treinamento, previsores_teste, classe_treinamento,
classe_teste = train_test_split(previsores, classe, test_size=0.25,
random_state=0)

# Importação e treinamento da máquina.
from sklearn.svm import SVC
classificador =  SVC(kernel = 'linear', random_state = 1)
classificador.fit(previsores_treinamento, classe_treinamento)

# Teste da máquina, realização das predições.
previsoes = classificador.predict(previsores_teste)

# Importação de bibliotecas para extrair métricas de performance do
algoritmo.
from sklearn.metrics import confusion_matrix, accuracy_score,
roc_auc_score, classification_report
precisao = accuracy_score(classe_teste, previsoes)
matriz = confusion_matrix(classe_teste, previsoes)

# Mostra na tela as métricas e resultados do algoritmo.
print(classification_report(classe_teste, previsoes))
print(matriz)
```

Apêndice F – Código fonte do algoritmo *Naive Bayes* com PCA

```
# Importação da biblioteca pandas, para a leitura do arquivo CSV.
import pandas as pd

# Leitura da base de dados e atribuindo para a variável "Base"
base = pd.read_csv('creditcard.csv')

# Atribui para a variavel "previsores" as colunas/variáveis da
aplicação que no caso são 30 colunas.
previsores = base.iloc[:, 0:30].values

# Atribui para a variável "classe" a coluna/variável o rótulo dos
registros.
classe = base.iloc[:, 30].values

# Aplicação do PCA
from sklearn.decomposition import PCA
pca = PCA(n_components=5)
previsores = pca.fit_transform(previsores)

# Importação da função "train_test_split" que irá dividir a base de
dados em uma base de treinamento e outra para teste.
from sklearn.model_selection import train_test_split
previsores_treinamento, previsores_teste, classe_treinamento,
classe_teste = train_test_split(previsores, classe, test_size=0.25,
random_state=0)

# Importação e treinamento da máquina.
from sklearn.naive_bayes import GaussianNB
classificador = GaussianNB()
classificador.fit(previsores_treinamento, classe_treinamento)

# Teste da máquina, realização das predições.
previsoes = classificador.predict(previsores_teste)

# Importação de bibliotecas para extrair métricas de performance do
algoritmo.
from sklearn.metrics import confusion_matrix, accuracy_score,
roc_auc_score, classification_report
precisao = accuracy_score(classe_teste, previsoes)
matriz = confusion_matrix(classe_teste, previsoes)

# Mostra na tela as métricas e resultados do algoritmo.
print(classification_report(classe_teste, previsoes))
print(matriz)
```