ALEXANDRE XAVIER CIUFFATELLI
PEDRO MIGUEL MACHADO GONÇALVES

CLUSTERIZAÇÃO NO ESPAÇO DE LATÊNCIA DE UMA REDE NEURAL

São Paulo
2020

ALEXANDRE XAVIER CIUFFATELLI
PEDRO MIGUEL MACHADO GONÇALVES

CLUSTERIZAÇÃO NO ESPAÇO DE LATÊNCIA DE UMA REDE NEURAL

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro de Computação.

São Paulo
2020

ALEXANDRE XAVIER CIUFFATELLI
PEDRO MIGUEL MACHADO GONÇALVES

CLUSTERIZAÇÃO NO ESPAÇO DE LATÊNCIA DE UMA REDE NEURAL

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro de Computação.

Área de Concentração:
Engenharia de Computação

Orientador:
Prof. Dr. Jorge Luis Risco Becerra

São Paulo
2020

AGRADECIMENTOS

Aos nossos pais pelo apoio incondicional.

Ao professor Jorge Luis Risco Becerra pela orientação deste trabalho.

Ao Grupo Turing pelas discussões e aprendizados na área de Inteligência Artificial.

RESUMO

Os algoritmos de clusterização são amplamente utilizados ao se analisar um conjunto de dados. Porém, em vários cenários do mundo real encontramos diversos tipos de dados, como categóricos, ordinais, contínuos, imagens, séries temporais e outros e, como os algoritmos de clusterização se baseiam na distância entre pontos, eles só conseguem uma boa performance quando trabalham unicamente com dados numéricos. Além disso, esses algoritmos de clusterização não consideram a correlação entre os atributos e a variável objetivo, ou seja, não são otimizados para o cenário problema. O objetivo deste trabalho é expor uma técnica de clusterização que consegue trabalhar com diversos tipos de dados, além de ser otimizada para o cenário de aplicação. Essa técnica é utilizar a clusterização no espaço de latência de uma rede neural. Em vista disso, a proposta é apresentar a técnica, compará-la com o K-means, um dos algoritmos mais usados, e analisar seus resultados em diferentes situações. Para demonstrar a técnica, são utilizados três cenários problemas, cada um com aumento de complexidade. Dois deles são gerados artificialmente e o último é um problema com dados reais no cenário de empréstimo. Em cada cenário, as limitações do K-means são observadas em um conjunto de dados com atributos numéricos e não numéricos, onde os dados são particionados primeiro pelas variáveis categóricas. Em contrapartida, a técnica proposta, além de não ter essa limitação, gera grupos otimizados para a variável objetivo. Assim, ao analisar os resultados no cenário real, diversas oportunidades de melhoria podem ser mapeadas para gerar valor ao negócio.

Palavras-Chave – Clusterização, K-means, Rede Neural Artificial, espaço de latência, *feature vector*, empréstimos.

ABSTRACT

Clustering algorithms are widely used when analyzing a data set. However, in real-world scenarios, where there are several types of data, such as numeric and categorical, these algorithms do not achieve a good performance since they are based on the distance between points. Furthermore, clustering algorithms do not consider the correlation between attributes and the objective variable, that is, they are not optimized for a problem scenario. This paper aims to expose a clustering technique that can perform with different types of data and strive to be optimized for the application scenario. This technique is to use clustering algorithm in the latency space of a neural network. Therefore, the approach is to display the technique, compare it with K-means, one of the most commonly used algorithms, and analyze its results in different situations. In order to demonstrate the technique, three problem scenarios will be used, each one with an increase in the complexity. Two of them are artificially generated and the last is a problem with real data in the loan scenario. In each scenario, K-means limitations are observed in a data set with numeric and non-numeric attributes, where the data points are partitioned first by categorical variables. In contrast, the proposed technique, besides not having this limitation, generates groups optimized for the target variable. Hence, when analyzing the results in the real scenario, several improvement opportunities can be mapped to generate business value.

Keywords – Clustering, K-means, Artificial Neural Network, Latency Space, feature vector, loan.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE CÓDIGOS

LISTA DE ABREVIATURAS E SIGLAS

WCSS	*Within-Cluster-Sum-of-Squares*
AE	*Autoencoder*
PCA	*Principal Component Analisys*
PC	*Principal Component*
CRISP-DM	*Cross-industry standard process for data mining*
P2P	*Peer-to-peer*
LC	*Lending Club*

SUMÁRIO

1 INTRODUÇÃO

Neste capítulo serão abordados os conceitos iniciais deste trabalho. Primeiro a motivação, seguido do objetivo, da metodologia e, por fim, sua organização.

1.1 Motivação

Ao analisar um conjunto de dados, tem-se como objetivo descrever os atributos e seus comportamentos. Para isso, exploramos sua estrutura, inferindo correlações, detectando anomalias, identificando características marcantes, semelhanças e dissimilaridades entre os dados. Isso nos permite obter *insights* e gerar hipóteses que ajudam a entender o problema e buscar soluções.

Neste contexto, os algoritmos de clusterização são amplamente utilizados. A ideia base é encontrar grupos em que elementos do mesmo grupo devem apresentar uma alta similaridade, mas devem ser dissimilares de objetos de outros grupos. A grande vantagem dessa ferramenta é que podemos descrever as peculiaridades de cada um dos grupos, aumentando o entendimento do conjunto de dados. Estes algoritmos são estudados e aprimorados até hoje [1].

Como exemplos de aplicações práticas, podemos citar: reconhecimento de padrões de consumo ou compra, identificação de perfis de clientes em crédito ou marketing, estudos de dados de genoma, entre outros.

Em vários cenários do mundo real encontramos diversos tipos de dados, como categóricos, ordinais, contínuos, imagens, séries temporais e outros. Porém, como os algoritmos de clusterização se baseiam na distância entre pontos, eles só conseguem uma boa performance quando trabalham unicamente com dados numéricos. Embora pesquisas sugiram técnicas para solucionar este problema, na maioria das vezes transformando os dados categóricos em numéricos antes da clusterização [2], os resultados ainda mostram um espaço para melhora.

Além disso, os algoritmos de clusterização não consideram a relação dos atributos com uma variável *target*, ou seja, não são otimizados para o cenário problema.

Assim, os algoritmos de clusterização não conseguem uma boa performance quando se misturam dados numéricos com não numéricos e os clusters encontrados não são otimizados para o problema abordado, o que dificulta encontrar boas soluções para os cenários complexos que temos no mundo real.

1.2 Objetivos

Este trabalho pretende expor uma técnica que tem como resultado final clusters, sendo, portanto, um método de clusterização. Ela consegue performar com diversos tipos de dados, como categóricos, numéricos e ordinais, e é aplicada em problemas que tenham pelo menos uma variável *target*. Como exemplos de aplicações desta técnica, temos: reconhecimento de padrões de consumo ou compra, identificação de perfis de clientes em crédito ou de grupos de risco para uma doença.

Para realizar a clusterização, o modelo usa como base as distâncias entre os *feature vectors* de um espaço de latência gerado a partir de uma rede neural treinada. Ou seja, esta técnica passa por uma etapa de treinamento supervisionado, com um treinamento de uma rede neural, e depois por uma não supervisionada, com a clusterização no espaço de latência.

Para atingir esse objetivo, utilizaremos diferentes cenários, mostrando o funcionamento passo a passo da abordagem e resultados obtidos, além de comparar com modelos tradicionais de clusterização.

Dessa forma, os objetivos específicos são definir a técnica, expô-la em cenários artificiais, comparando com modelos tradicionais, e mostrar sua eficácia em problemas complexos e reais.

1.3 Justificativa

Dentre as técnicas de clusterização, existe uma que consiste em treinar uma rede neural autoencoder (AE), segmentar a rede e utilizar os *embeddings* gerados na camada de latência para realizar a clusterização [3].

Outra técnica que utiliza a segmentação de uma rede neural treinada é a de reconheci-

mento facial, como é o exemplo da rede *DeepFace* [4]. Nela, uma rede neural convolucional é treinada para, dado uma imagem de um rosto, identificar a quem pertence aquela face. Depois, é utilizado o vetor de saída da última camada escondida, chamada de *raw face representation feature vector* [1], para comparar a distância de um vetor gerado a partir de um rosto conhecido com o de um outro desconhecido. Neste exemplo, o vetor representa o conceito de um rosto em um espaço euclidiano no qual a rotação e o fundo da imagem não impactam de maneira considerável o resultado, mas características faciais como distância entre os olhos, boca e nariz influenciam drasticamente.

Dado que este vetor está em um espaço euclidiano, gerado por uma rede neural, podemos comparar esse espaço com o espaço de latência do AE, em que dados similares estão próximos nesse espaço.

Tendo em mente que o estado da arte de clusterização consiste em transformar os dados de entrada numéricos e não numéricos, por meio de um processamento [2], exclusivamente em dados numéricos para depois utilizar um algoritmo de clusterização, e que o vetor gerado a partir de uma rede neural treinada é exclusivamente numérico, além de representar um conhecimento aprendido pelo modelo, é proposto realizar a clusterização no espaço de latência gerado por uma rede neural treinada para um domínio, ou seja, com uma variável *target*.

1.4 Metodologia

1.4.1 Pesquisa bibliográfica

Para entender o estado da arte de clustering, procurou-se artigos referências na área. Além disso, a pesquisa bibliográfica se fez necessária para embasar a técnica proposta e para confirmar os conhecimentos preliminares.

O conhecimento sobre mercado financeiro veio a partir da experiência na área de crédito, especificamente em empréstimos, que ambos os integrantes possuem, complementada por pesquisas sobre o assunto.

1.4.2 Determinação do problema

A definição do tema se deu por meio da pesquisa bibliográfica e experiência dos integrantes do grupo, que participaram de grupo de extensão focado em inteligência artificial e

[1] vetor bruto da representação das características faciais

possuem experiência profissional nesta área. A partir disso, foram levantadas as limitações dos algoritmos tradicionais de clusterização.

1.4.3 Estruturação da solução

Para estruturar a solução, a técnica sugerida será descrita passo a passo, com o embasamento em artigos, assim como seus cenários de aplicação.

1.4.4 Desenvolvimento e Teste

Para testar a técnica, pretende-se utilizá-la em dois cenários gerados artificialmente e compará-la com um modelo tradicional de clusterização K-means. Ainda, a técnica será aplicada em um cenário real com dados fornecidos pelo *Kaggle* [5].

1.4.5 Elaboração da Monografia

A monografia foi redigida em LaTeX à medida que se teve progresso no desenvolvimento da solução.

1.5 Organização do Trabalho

A organização do trabalho foi dividida em cinco capítulos. O primeiro capítulo, ao qual esta seção pertence, introduz o tema, assim como sua motivação e metodologia.

O capítulo dois apresenta os conceitos técnicos utilizados, com foco em processamento de dados e aprendizado de máquina.

No capítulo três será exposta a técnica de clusterização proposta e o planejamento de desenvolvimento do trabalho.

O quarto capítulo demonstra o desenvolvimento nos cenários propostos e os resultados obtidos.

O quinto e último capítulo consiste nas considerações finais, e contém as conclusões, contribuições e as perspectivas de continuações do trabalho.

2 ASPECTOS CONCEITUAIS

Nesta seção, vamos expor os conceitos teóricos utilizados.

2.1 Clusterização

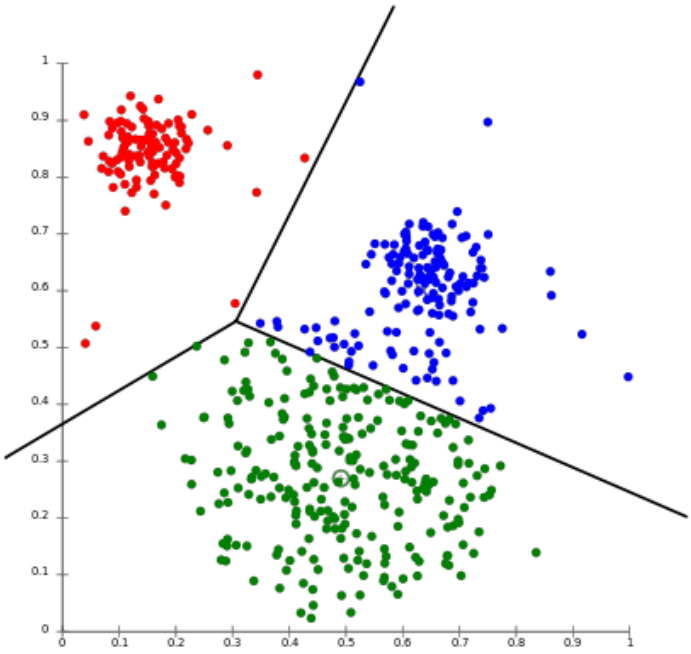
Clusterização [1] é uma técnica de *machine learning* não supervisionada usada para agrupar dados não rotulados em clusters que contêm pontos de dados 'semelhantes' um ao outro e 'diferentes' dos de outros clusters. Muitos algoritmos de *clustering* podem lidar apenas com dados que contêm valores de recurso numéricos ou categóricos. Os recursos numéricos podem assumir valores reais, como altura, peso e distância. Os recursos categóricos representam dados que podem ser divididos em um número fixo de categorias, como cor, raça, sexo, profissão e grupo sanguíneo. Os algoritmos de clusterização agrupam os pontos de dados em clusters usando alguma noção de 'similaridade', que pode ser tão simples quanto a distância euclidiana. Para calcular a semelhança entre os valores de recurso numéricos, operações matemáticas (como distâncias, ângulos, soma ou média) são aplicadas a eles.

As medidas de similaridade com base na distância são usadas principalmente para pontos de dados numéricos. Geralmente, os valores das características categóricas não são inerentemente ordenados (por exemplo, os valores categóricos engenharia, medicina e direito). Não é possível calcular diretamente a distância entre dois valores de recursos categóricos. Portanto, calcular medidas de similaridade baseadas em distância para dados categóricos é uma tarefa desafiadora. No entanto, vários métodos têm sido sugeridos na literatura para calcular a similaridade entre pontos de dados contendo características categóricas.

2.1.1 K-Means

O K-means [6] [7] é um método de agrupamento que particiona um conjunto de pontos de dados em K grupos diferentes, e cada um é formado por um centroide e seus pontos mais próximos, como exemplificado na figura 4.

Figura 1: Exemplo K-Means



Fonte: AWS Machine Learning Blog, 2018

O algoritmo parte de um número K, que é o número de clusters. A partir dele, são gerados K pontos aleatórios e definidos como centroides. Os pontos são atribuídos ao centroide mais próximo e, juntos, formam um cluster. Depois, os centroides são ajustados para o ponto médio do seu cluster. Dessa forma, os pontos são novamente atribuídos aos centroides mais próximos, formando novos clusters, e a posição dos centroides são reajustadas. Esses passos são repetidos até que todos os pontos estejam perfeitamente organizados, ou seja, os centroides se mantêm no mesmo ponto, ou um número máximo de iterações seja atingido.

2.1.2 Escolhendo o número correto de clusters

O número de clusters que escolhemos para um determinado conjunto de dados não pode ser aleatório. Uma maneira simples de descobrir o número certo de clusters é a partir

do *Within-Cluster-Sum-of-Squares* (WCSS). Nele, cada cluster é formado pelo cálculo e comparação das distâncias dos pontos de dados dentro de um cluster com seu centroide.

WCSS é a soma dos quadrados das distâncias de cada ponto de dados em todos os clusters até seus respectivos centroides, como mostrado na equação abaixo:

$$\sum_{C_k}^{C_n}(\sum_{d_i in C_i}^{d_m} distance(d_i, C_k)^2) \tag{2.1}$$

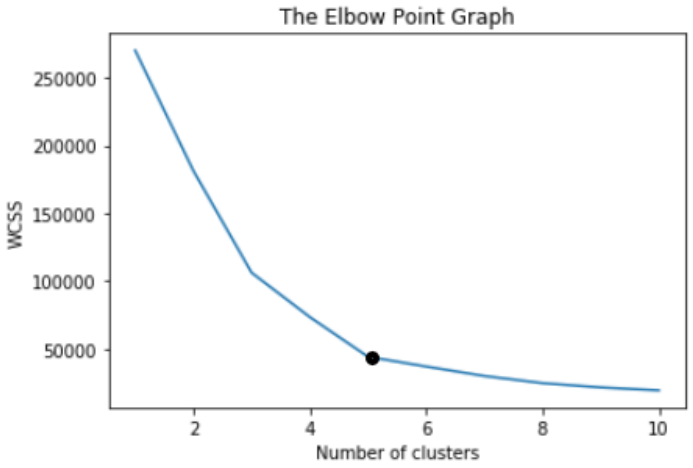
Onde C é um centroide dos clusters e d é um ponto dentro de cada cluster

A ideia é que, quanto mais clusters, menor é o valor dessa soma. Suponha que haja n observações em um determinado conjunto de dados e especificamos n número de clusters ($k = n$), então WCSS se tornará zero, uma vez que os próprios pontos de dados agirão como centroides e a distância será zero e, idealmente, isso forma um cluster perfeito. No entanto, não faz sentido, pois temos tantos clusters quanto as observações. Portanto, existe um valor limite para K que podemos encontrar a partir do WCSS, chamado de método do cotovelo (*Elbow Method*).

2.1.3 *Elbow Method*

Podemos encontrar o valor ótimo para K usando um gráfico de *Elbow*. Para um intervalo de valores de K, aplicamos o K-means e calculamos o WCSS e plotamos o gráfico do WCSS para cada valor de K. O gráfico resultante seria parecido com o da figura 2 abaixo:

Figura 2: Exemplo *Elbow Method*



Fonte: *Analytics India Magazine*, 2019

Podemos ver que o valor de WCSS diminui com um aumento no número de clusters. Selecionamos o valor ideal de K com base na taxa de diminuição do WCSS. Por exemplo, do cluster 1 ao 2 ao 3 no gráfico acima, vemos uma queda repentina e enorme no WCSS. Após 5, a queda é mínima e, portanto, escolhemos 5 como o valor ideal para K.

2.1.4 Avaliação dos clusters

A avaliação tem por objetivo inferir a qualidade dos clusters encontrados. Podemos fazer uma avaliação qualitativa, em que consideramos o quanto os clusters encontrados ajudaram na resolução do problema, ou quantitativa, na qual metrificamos os clusters formados a partir de fórmulas. Esta última é dividida em dois grupos, a de critério interno e externo, cada um com diversas métricas próprias.

As avaliações de critério internos têm como base os atributos utilizados para realizar a clusterização. Embora possuam a vantagem de que não é necessário conhecer os grupos esperados, elas atribuem uma melhor pontuação para resultados em que clusters possuem uma alta similaridade intra-cluster e dissimilaridade inter-cluster. Isso favorece os algoritmos baseados em distância, como o K-means.

Outro tipo de avaliação é a que considera critérios externos, ou seja, se baseiam em dados que não foram usados para realizar a clusterização, como uma classe ou agrupamento esperado. Neste trabalho, utilizamos a métrica de pureza [7], definida a seguir.

2.1.4.1 Pureza

Esta métrica não precisa de um agrupamento predeterminado, mas utiliza classes conhecidas. Ela é definida pela seguinte fórmula:

$$purity(\Omega, \mathbb{C}) = \frac{1}{N} \sum_{k} \underset{j}{max} |w_k \cap c_j| \tag{2.2}$$

sendo $\Omega = \{\omega_1, \omega_2, ..., \omega_K\}$ o conjunto de clusters, $\mathbb{C} = \{c_1, c_2, ..., c_J\}$ o conjunto de classes e N o número total de pontos.

A fórmula pode ser interpretada da seguinte maneira: para cada grupo, conta-se a frequência de cada classe presente, então soma-se as frequências mais observadas em cada cluster e divide-se pelo número de pontos totais. Dessa forma, se cada cluster contiver apenas uma única classe, a purezá será 1, o valor máximo.

2.2 Redes Neurais

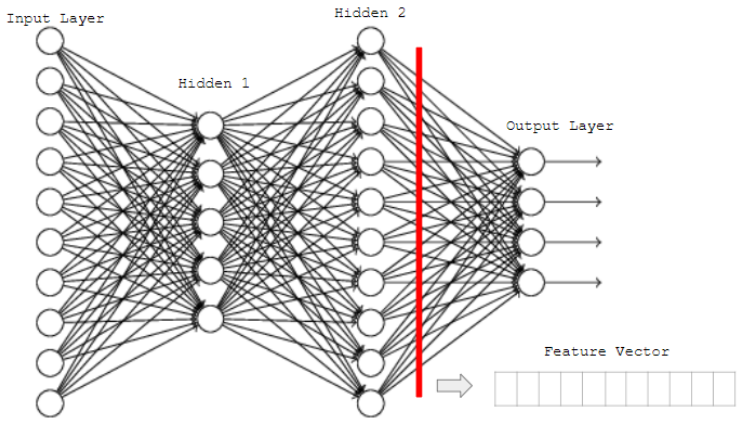
Rede neural artificial é um algoritmo inspirado no funcionamento de um cérebro, mais especificamente de uma rede de neurônios, e tem como objetivo reconhecer padrões. Tais algoritmos aprendem a reconhecer padrões por meio de exemplos sem a necessidade de se programar tarefas específicas. Os padrões que elas reconhecem são numéricos, contidos em vetores, nos quais todos os dados do mundo real devem ser traduzidos em números, sejam imagens, som, texto ou séries temporais. Este método de *machine learning* se popularizou por sua alta capacidade de generalização.

2.2.1 Reconhecimento Facial

Reconhecimento Facial é uma técnica que consiste em identificar, a partir de uma foto de um rosto, a quem aquele rosto pertence. Devido à forte capacidade de generalização, redes neurais conseguem realizar essa tarefa com uma alta precisão.

Esse é o caso da rede DeepFace [4]. Nela, uma rede neural é treinada para identificar uma pessoa conhecida a partir de uma imagem com seu rosto. Uma vez que temos este modelo treinado, a rede é segmentada e a nova saída se torna o vetor gerado na penúltima camada. Essa nova saída é chama de *feature vector*, como exemplificado na imagem 3.

Figura 3: *Feature Vector* de uma Rede Neural



Fonte: Autoria própria

Idealmente, como o modelo aprendeu a identificar rostos, cada rosto gera um vetor único, e um mesmo rosto em fotos diferentes gera o mesmo vetor. Na prática, um mesmo rosto em fotos diferentes gera vetores próximos. Assim, sabendo que temos um vetor u gerado a partir da pessoa A, dado um novo rosto de uma pessoa desconhecida B,

calculamos o vetor v deste novo rosto. Se a distância entres os vetores u e v for próxima, dizemos que os rostos A e B são da mesma pessoa, caso contrário, dizemos que o rosto B não é da pessoa A. Ou seja, utilizamos a distância entre os vetores gerados para realizar a identificação facial.

Neste exemplo, o *feature vector* representa o conceito de um rosto em um espaço de latência onde a rotação, luminosidade e o fundo da imagem não impactam de maneira considerável o resultado, mas características faciais como distância entre os olhos, boca e nariz influenciam drasticamente.

2.2.1.1 *Feature vector*

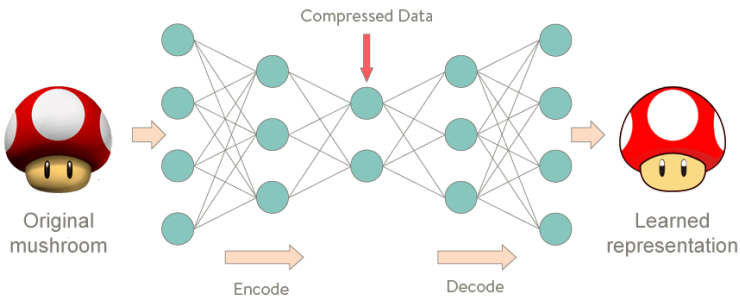
Um *feature vector* (vetor característico) é um vetor n-dimensional que possui as características de um objeto representadas numericamente. No contexto de redes neurais, ele é gerado a partir de uma rede e carrega as características aprendidas pela rede. Esse vetor não tem um significado físico.

2.2.2 Autoencoder

Autoencoder [8] é uma arquitetura de rede neural em que a saída da rede é a mesma que sua entrada e, ao longo da rede, a entrada é comprimida e depois reconstruída. Seu objetivo é obter uma representação compacta da entrada. Para isso, ela conta com duas partes.

A primeira é o codificador (*encoder*), que codifica a entrada em uma representação com menos dimensões em um espaço de latência. A entrada comprimida é chamada de *embedding*. A segunda é o decodificador (*decoder*), que reconstroi a entrada a partir do embedding gerado. Esse processo pode ser visto na figura abaixo.

Figura 4: *Mushroom* Encoder



Fonte: *Deep Learning Book*, Capítulo 58

2.2.2.1 Espaço de Latência

Espaço de latência, do Latin: *lateo* ("escondido"), é um espaço gerado por um modelo matemático a partir de um espaço conhecido com variáveis observáveis. Ou seja, ele por si só não possui um significado físico, por isso é conhecido como um espaço "escondido".

Em *machine learning*, este espaço [9] é comumente utilizado para representar entradas em um espaço mais conveniente. Neste espaço, entradas parecidas geram vetores próximos. Um exemplo clássico é a projeção de fotos com faces em um espaço onde conseguimos realizar a identificação facial por proximidade dos vetores gerados. Outro exemplo é o espaço gerado por uma rede autoenconder, onde o espaço é capaz de representar entradas complexas em vetores menores.

2.2.2.2 *Embedding*

Um *embedding* é uma representação vetorial em baixa dimensão gerado a partir de um espaço de alta dimensão.

2.3 Preparação de dados

2.3.1 Codificação *One-hot*

One hot é uma maneira de transformar uma coluna categórica em colunas binárias por categoria, em que 0 significa não pertencer aquela categoria e 1 significa pertencer. Um exemplo pode ser visto na figura 5 abaixo, onde a coluna categória 'Curso' é tranformada nas colunas binárias 'Curso_direito', 'Curso_engenharia', 'Curso_medicina'.

Figura 5: Exemplo de transformação *one-hot*

id_aluno	Curso
1	engenharia
2	medicina
3	direito
4	engenharia
5	medicina

id_aluno	Curso_direito	Curso_engenharia	Curso_medicina
1	0	1	0
2	0	0	1
3	1	0	0
4	0	1	0
5	0	0	1

Fonte: Autoria própria

2.3.2 Redução de dimensionalidade

Redução de dimensionalidade é a transformação de dados de um espaço de alta dimensão em um espaço de baixa dimensão para que a representação de baixa dimensão retenha algumas propriedades significativas dos dados originais, idealmente perto de sua dimensão intrínseca.

Os métodos são comumente divididos em abordagens lineares e não lineares. As abordagens também podem ser divididas em seleção e extração de recursos. A redução de dimensionalidade pode ser usada para redução de ruído, visualização de dados, análise de cluster ou como uma etapa intermediária para facilitar outras análises. Dentre os métodos, a Análise de Componentes Principais é um dos mais utilizados.

2.3.2.1 *Principal Component Analysis*

A Análise de Componentes Principais (PCA) é um procedimento estatístico que transforma ortogonalmente as n coordenadas originais de um conjunto de dados em um novo conjunto de n coordenadas chamadas componentes principais.

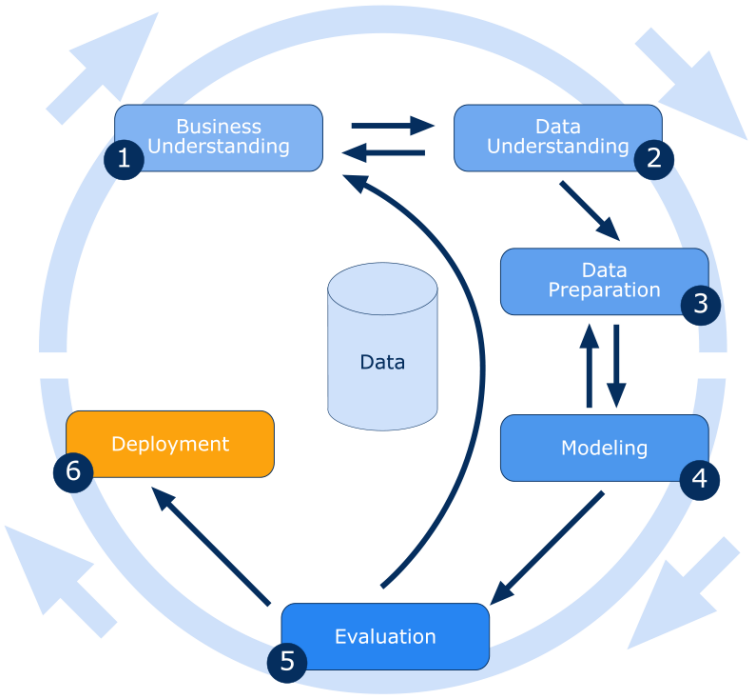
Como resultado da transformação, o primeiro componente principal tem a maior variação possível. Cada componente sucessivo tem a maior variação possível sob a restrição de que é ortogonal (ou seja, não correlacionado com) aos componentes anteriores. Manter apenas os primeiros m componentes, com $m < n$, reduz a dimensionalidade dos dados enquanto retém a maioria das informações dos dados, ou seja, a variação nos dados.

É importante notar que a transformação PCA é sensível à escala relativa das variáveis originais. Os intervalos da coluna de dados precisam ser normalizados antes de aplicar o PCA. Além disso, as novas coordenadas (PCs) não são mais variáveis reais produzidas pelo sistema. Aplicar o PCA ao seu conjunto de dados perde sua interpretabilidade. Aplicar o PCA no seu conjunto de dados acarreta a perda da sua interpretabilidade.

2.4 CRISP-DM

O CRISP-DM (*Cross-industry standard process for data mining*) [10] é uma metodologia que organiza um processo de mineração de dados em seis fases: entendimento do modelo de negócio, entendimento dos dados, preparação dos dados, modelagem, avaliação e implementação.

Figura 6: CRISP-DM



Fonte: Fonte: *Semantix Blog*

Esse metodo é amplamente utilizado devido à sua poderosa praticidade, flexibilidade e utilidade ao usar a análise para resolver problemas de negócio complexos.

3 ESPECIFICAÇÃO DA CLUSTERIZAÇÃO NO ESPAÇO DE LATÊNCIA

3.1 Contexto de aplicação

A técnica proposta tem como aplicação um cenário em que se busca grupos voltados para um problema específico. Em outras palavras, os grupos encontrados são otimizados para uma variável *target*. Por conta disso, uma limitação é que ela só pode ser aplicada quando tivermos pelo menos uma variável *target*.

O fato da técnica depender de uma rede neural traz vantagens e desvantagens. Como vantagem temos que, enquanto os algoritmos tradicionais de clusterização não conseguem uma boa performance quando trabalham com dados não numéricos, a técnica proposta não apresenta essa limitação por utilizar uma rede neural para processar as entradas. Isso é, na verdade, uma grande vantagem, visto que diversos problemas do mundo real apresentam dados numéricos e não numéricos como entrada. Outra vantagem é que se consegue aproveitar a forte capacidade de generalização da rede neural. Porém, a desvantagem é que o resultado fica diretamente atrelado à performance da rede e, portanto, necessita de uma rede com uma acurácia relevante, ou seja, que tenha um bom aprendizado do problema.

3.2 Estrutura da clusterização

A técnica proposta é a de aplicar o algoritmo de clusterização em um espaço de latência gerado por uma rede neural treinada para um domínio específico, em contraposição à abordagem tradicional de aplicar o algoritmo diretamente no vetor de entrada. Ao longo desta seção, será exposto com mais detalhes a técnica e o embasamento teórico por trás dessa ideia.

Inspirado no modelo de reconhecimento facial DeepFace [4], observamos que o espaço de latência na última camada escondida gerado por uma rede neural treinada representa

o conhecimento aprendido por meio de um *feature vector*. Neste espaço, os dados de entrada relevantes para o problema possuem um peso maior, além de gerar um vetor exclusivamente numérico a partir de uma entrada, esta podendo conter dados mistos.

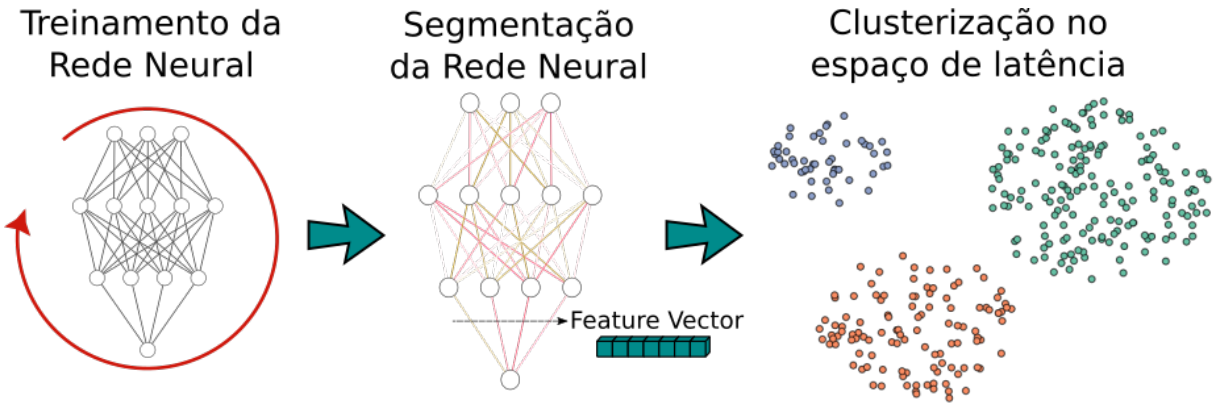
Então, dado que o vetor de entrada está em um espaço V, e o vetor de latência é um espaço em W, podemos escrever a operação da rede como sendo a transformação:

$$T : V \rightarrow W \tag{3.1}$$

Propõe-se, portanto, aplicar um algoritmo de clusterização no espaço W gerado pela rede neural, em contraposição a aplicar diretamente nos dados de entrada em V. Dessa forma, é possível tirar vantagem do forte poder de generalização das redes neurais e da capacidade de performar bem com diferentes tipos de entrada. Essa técnica, portanto, exige uma etapa de treinamento supervisionado.

Assim, como mostrado na figura 7, a primeira etapa é treinar uma rede neural para uma variável *target* de interesse do cenário problema. Após o treinamento, segmentamos a rede extraindo o espaço de latência para criar os *feature vectors* das nossas entradas. Por fim, utilizamos o k-means nos *feature vectors* para encontrar os clusters.

Figura 7: Etapas para clusterização no espaço de latência



Fonte: Autoria própria

3.3 Desenvolvimento

3.3.1 Um projeto de aprendizagem de máquina

Para maximixar o sucesso da aplicação, o projeto de aprendizado de máquina deve seguir um planejamento. Isto porque o modelo em si não é o resultado final, mas sua aplicação ou as conclusões tiradas a partir dele são. Portanto, seguimos a metodologia CRISP-DM e dividimos o projeto nas seguintes etapas:

1. Entendimento do cenário
2. Exploração dos dados
3. Preparação dos dados
4. Treinamento de modelos
5. Análise dos resultados

3.3.1.1 Entendimento do cenário

Nesta etapa vamos definir qual o cenário, o objetivo esperado e o planejamento.

3.3.1.2 Exploração de dados

Vamos explorar quais dados compõem o *dataset*, suas distribuições, as correlações entre si e com a variável *target*. O objetivo é mostrar quais são as variáveis disponíveis e como elas se comportam.

Além disso, durante esta etapa, levantaremos hipóteses sobre o problema e buscaremos testá-las para extrair variáveis relevantes. Este processo é chamado de *feature engineering*.

3.3.1.3 Preparação dos dados

Após a exploração de dados, já temos um conjunto de variáveis que podemos utilizar como entrada do modelo. Nesta etapa vamos organizar a extração desse conjunto de *features*, fazendo o pré processamento necessário.

Tanto para um algoritmo de clusterização quanto para uma rede neural, uma boa prática é normalizar ou escalonar os dados. Além disso, as colunas com dados categóricos

serão transformadas em uma coluna binária por categoria, com 0 para não pertence à categoria e 1 para pertence.

Ainda, é necessário dividir o conjunto de dados em 2 subconjuntos, um para treino no qual todas as variáveis podem ser utilizadas, e outro de validação, em que o modelo poderá usar apenas as variáveis de entrada. O objetivo do subconjunto de validação é verificar como o modelo treinado performa com dados completamente novos.

Vale ressaltar que a normalização ocorre primeiro nos dados de treino e, com os mesmos hiperparâmetros de valor máximo e mínimo, os dados de validação são transformados.

3.3.1.4 Treinamento do modelo

Nesta etapa vamos treinar o modelo e escolher seus hiperparâmetros.

3.3.1.5 Análise dos resultados

Na etapa final, o objetivo é verificar o resultado do modelo nos dados de validação e sua performance por meio de uma avaliação qualitativa e quantitativa.

3.3.2 Fases da implementação

Nesta seção vamos definir como será a implementação do trabalho.

3.3.2.1 Elaboração dos cenários

Para demonstrar a técnica de clusterização no espaço de latência, são utilizados três cenários problemas, em que a complexidade é progressivamente aumentada. Dois deles são gerados artificialmente e o último é um problema com dados reais no mercado de empréstimo. Todos os cenários possuem dados numéricos e categóricos.

O primeiro cenário possui um conjunto de dados simples, ou seja, as variáveis de entrada possuem uma correlação clara com a variável *target*, e separável, ou seja, existe uma maneira de inferir a variável *target* a partir das entradas com total acurácia.

Já no segundo cenário, o conjunto de dados gerado é simples, mas não é completamente separável. Isto porque existe uma parte do conjunto em que não é possível inferir a variável *target* com total acurácia a partir das entradas.

No terceiro e último cenário, o conjunto de dados é complexo e não separável. Ele é

composto por um conjunto de dados reais, retirados do site de competições Kaggle [5], e fornecidos pela empresa Lending Club.

3.3.2.2 Aplicação dos modelos

Cada cenário é independente e, portanto, cada um será tratado como um projeto e seguirá o planejamento proposto na seção 3.3.1. Para cada um, realizaremos a clusterização com o K-means no espaço de latência e diretamente nas entradas para comparação de performance.

3.3.2.3 Análise de resultados

Faremos uma avaliação qualitativa, em que consideramos o quanto os clusters encontrados ajudaram na interpretação do cenário ou na resolução do problema, e outra quantitativa, baseado em critérios externos. A métrica escolhida é a de pureza por não precisar do agrupamento predeterminado.

Para esta forma de avaliação, não utilizamos critérios internos pois comparamos a clusterização realizada em espaços diferentes, onde as distâncias não têm o mesmo significado.

3.3.3 Tecnologias utilizadas

Neste trabalho vamos utilizar ferramentas voltadas para se trabalhar com dados e *machine learning*.

3.3.3.1 Framework de desenvolvimento

O *Jupyter Notebook* [11] é uma aplicação web que permite a criação e compartilhamento de documentos que tenham texto, imagens, códigos. É muito utilizada para a manipulação de dados pois permite a execução de código em fragmentos. Por isso, utilizaremos essa ferramenta para desenvolver as soluções.

3.3.3.2 Linguagem de programação

A linguagem de programação utilizada é *Python 3* [12]. Ela foi escolhida por ser versátil, ter uma sintaxe simples, ter uma boa legibilidade e pela nossa experiência prévia. Complementar a essa linguagem, utilizamos as bibliotecas listadas a seguir.

3.3.3.3 Bibliotecas

Scikit-learn formalmente conhecido como *scikits.learn*, e também conhecido como *sklearn*, é uma biblioteca de *machine learning* e de software livre para a linguagem de programação *Python*. Possui diversos algoritmos prontos, desde pré processamento até algoritmos de classificação, regressão e clusterização. [13]

NumPy é a bibliteca otimizada em C para manipular *arrays n* dimencionais. [14]

Pandas utilizado para manipulação e análise de dados. Em particular, oferece estruturas e operações de dados para manipulação de tabelas e séries temporais. [15]

Matplotlib uma biblioteca de visualização de dados por meio de gráficos. [16]

TensorFlow é uma biblioteca de software livre e de código aberto para fluxo de dados e programação diferenciável. É uma biblioteca matemática simbólica, e também é usada para aplicações de *machine learning*, como redes neurais. É usada para pesquisa e produção no Google. [17]

4 IMPLEMENTAÇÃO

4.1 Cenário 1

Para ilustrar o funcionamento da técnica de clusterização no espaço de latência, será utilizado um conjunto de dados gerados artificialmente, formado a partir de regras. Este cenário tem um fim didático, buscando explorar o comportamento da técnica em um domínio simples e separável.

O objetivo é expor a técnica, como ela funciona, seu resultado e comparar com o K-means aplicado diretamente nas entradas. Para isso, primeiro vamos apresentar os dados gerados, seguido de uma análise do conjunto de dados e da preparação para a modelagem. Depois, será realizado o treinamento dos modelos K-means aplicado diretemente nas entradas e do K-means aplicado no espaço de latência gerado por uma rede neural com a exposição dos resultados de cada abordagem. Por fim, vamos comparar os resultados obtidos pelas duas abordagens de clusterização.

4.1.1 Dados gerados

Os dados gerados possuem 3 variáveis de entrada e uma de saída. Cada variável segue uma distribuição definida por parâmetros. Essas variáveis são geradas a partir de regras, onde cada regra possui parâmetros próprios para das distribuições. O código gerador de dados utilizado foi:

Código 1: Gerador dos dados artificiais

```
def generate_data(data_name, data_size, param_1, param_2,
    param_3, out_param):
    """
       Generate a profile based on some distributions (rules)
       input: data name, sample size and the distribution
       parameters in dict format
       output: DataFrame with generated data
    """
```

```
    var_name = np.array([data_name] * data_size)

    # input variables
    var_1 = np.random.normal(loc=param_1['mu'],
                             scale=param_1['sigma'],
                             size=data_size).round(4)
    var_2 = np.random.uniform(param_2['inf_lim'],
                             param_2['sup_lim'],
                             data_size).round(4)
    var_3 = np.random.choice(param_3['choices'],
                             data_size,
                             p=param_3['weights'])

    # output variables
    output = np.array([out_param] * data_size)

    df = pd.DataFrame(data={'rule':var_name,
                            'var_1':var_1,
                            'var_2':var_2,
                            'var_3':var_3,
                            'output':output
                           })

    return df
```

Fonte: Autoria própria

Dessa forma, a primeira variável de entrada é numérica e segue uma distribuição normal com parâmetros média (*mu*) e desvio padrão (*sigma*), a segunda também é numérica e segue uma uniforme com parâmetros de limite inferior (*inf_lim*) e superior (*sup_lim*) e a terceira é categórica com parâmetros de categorias (*choices*) e suas respectivas frequências (*weights*). A saída é um valor único por regra.

Para este problema, utilizamos 3 regras, sendo cada uma definida como:

Código 2: Regras cenário 1

```
rule_1 = generate_data(
    data_name = 'rule_1',
    data_size = 3000,
    param_1 = {'mu':-1, 'sigma':0.1},
    param_2 =  {'inf_lim':-1, 'sup_lim':0},
    param_3 = {'choices':['Cat_1', 'Cat_2'],
            'weights':[0.5, 0.5]},
```

```
    out_param = 1
)

rule_2 = generate_data(
    data_name = 'rule_2',
    data_size = 3000,
    param_1 = {'mu':0, 'sigma':0.1},
    param_2 = {'inf_lim':-0.5, 'sup_lim':0.5},
    param_3 = {'choices':['Cat_2', 'Cat_3'],
            'weights':[0.5, 0.5]},
    out_param = 0
)

rule_3 = generate_data(
    data_name = 'rule_3',
    data_size = 3000,
    param_1 = {'mu':1, 'sigma':0.1},
    param_2 = {'inf_lim':0, 'sup_lim':1},
    param_3 = {'choices':['Cat_3', 'Cat_4'],
            'weights':[0.5, 0.5]},
    out_param = 1
)
```

Fonte: Autoria própria

O que resulta nos dados no seguinte formato:

Figura 8: Data Frame dos dados gerados no cenário 1

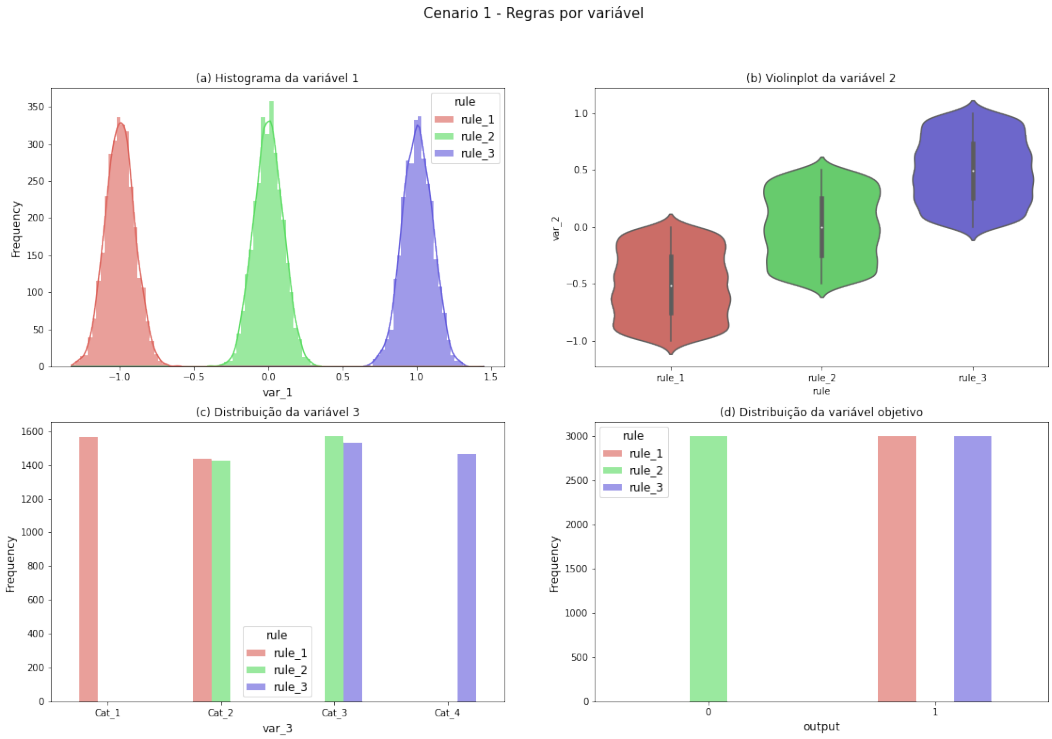
```
train.head()
```

id	var_1	var_2	var_3_Cat_1	var_3_Cat_2	var_3_Cat_3	var_3_Cat_4	output
3553	0.491270	0.685371	0.0	0.0	1.0	0.0	0.0
2772	0.110534	0.038960	1.0	0.0	0.0	0.0	1.0
417	0.121104	0.126182	0.0	1.0	0.0	0.0	1.0
8297	0.776407	0.674369	0.0	0.0	1.0	0.0	1.0
2391	0.083369	0.468867	0.0	1.0	0.0	0.0	1.0

Fonte: Autoria própria

Os dados gerados são mostrados na figura 9 a seguir, separados por variável e por regra.

Figura 9: Variáveis geradas no cenário 1 por regra



Fonte: Autoria própria

De acordo com o histograma (a), fica visível que a variável 1 é bem característica de cada regra, uma vez que não há intersecção entre elas. Assim, dizemos que este problema é separável uma vez que podemos diferenciar a variável de saída completamente pela variável 1.

Já no *violin plot* (b) da variável 2 vemos uma intersecção entre a regra 2 com as demais regras, porém, nos extremos inferior e superior estão presentes apenas as regras 1 e 3, respectivamente.

A variável 3 é uma variável categórica com quatro categorias. Pela figura (c), vemos que a categoria 1 está presente apenas na regra 1, já a categoria 2 é encontrada nas regras 1 e 2. A categoria 3 é observada nas regras 2 e 3, e por fim a categoria 4 aparece apenas na regra 3.

A variável final é uma variável de *output*. Vemos na figura (d) que ela tem valor 1 para as regras 1 e 3 e, para a regra 2, tem valor 0.

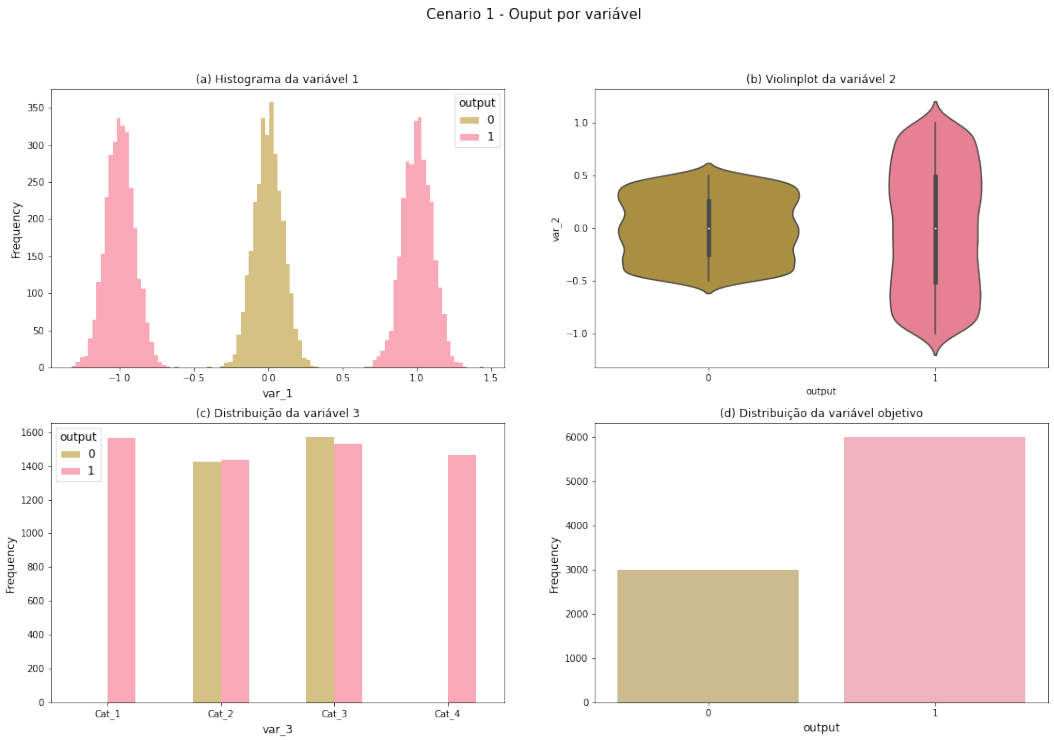
Vale ressaltar que um algoritmo de clusterização é não supervisionado e, portanto, não há uma única resposta correta. Ou seja, dizer que cada regra forma um cluster não

é, necessariamente, a resposta correta ou a melhor resposta.

4.1.2 Exploração dos dados

Ao olhar o conjunto de dados sem as regras, apenas sabendo das variáveis de entrada e *target*, temos a figura 10.

Figura 10: Distribuição das variáveis em relação ao output no cenário 1



Fonte: Autoria própria

A partir do histograma (a), percebemos que a variável *target* de saída 0 está concentrada no centro e que a de saída 1 está nas extremidades. Além disso, elas estão bem separadas, ou seja sem intercecções.

No *violin plot* (b), observamos que a variável 2 possui nos extremos exclusivamente a saída 1.

De acordo com o gráfico de barras (c), vemos que as categorias 1 e 4 possuem apenas a saída 1, enquanto as categorias 2 e 3 possuem ambas as saídas 0 e 1.

Pela distribuição (d), conclui-se que o conjunto de dados é desbalanceado, uma vez que a variável de saída 0 aparece bem menos que a 1.

4.1.3 Preparação dos dados

Para preparar os dados, o conjunto de dados será dividido em 2 subconjuntos, um para treino e outro de validação. Os subconjuntos de treino e validação serão os mesmos utilizados no modelo K-means diretamente nas entradas e K-means no espaço de latência.

Além disso, as colunas numéricas *var_1* e *var_2* serão normalizadas, e na coluna categórica *var_3* será aplicada a codificação one-hot, ou seja, transformada em uma coluna binária por categoria. O conjunto de treino fica com o seguinte formato:

Figura 11: Conjunto de treino cenário 1

```
train.head()
```

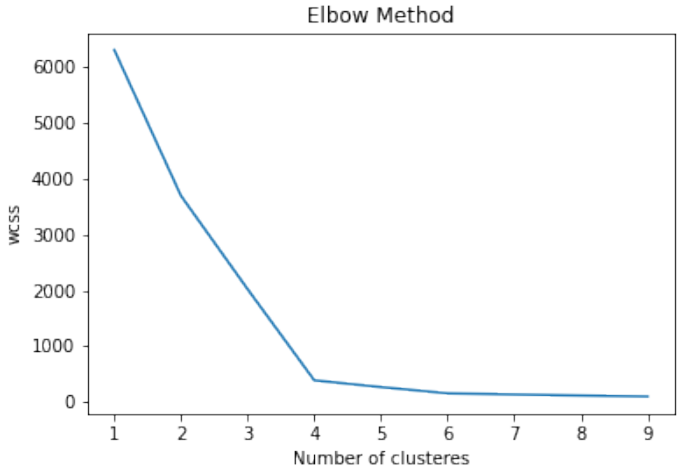
id	var_1	var_2	var_3_Cat_1	var_3_Cat_2	var_3_Cat_3	var_3_Cat_4	output
3553	0.491270	0.685371	0.0	0.0	1.0	0.0	0.0
2772	0.110534	0.038960	1.0	0.0	0.0	0.0	1.0
417	0.121104	0.126182	0.0	1.0	0.0	0.0	1.0
8297	0.776407	0.674369	0.0	0.0	1.0	0.0	1.0
2391	0.083369	0.468867	0.0	1.0	0.0	0.0	1.0

Fonte: Autoria própria

4.1.4 K-means nas entradas

Ao aplicar o K-means diretamente nas entradas, o primeiro parâmetro a ser definido é o número de clusters. Para tal definição, o *Elbow Method* auxilia. Ao gerar o gráfico da figura 12 abaixo, vemos que 4 é o número de clusters ideal. Para fins demonstrativos, vamos também forçar 6 clusters.

Figura 12: *Elbow Method* nas entradas no cenário 1



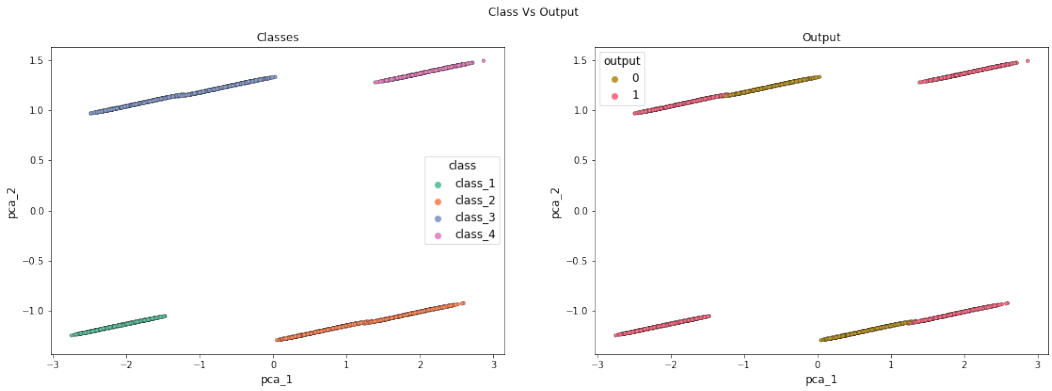
Fonte: Autoria própria

4.1.4.1 K-means nas entradas para k=4

Para visualizar os clusters, o método PCA será utilizado. Aplicando este método nas entradas, encontramos 2 componentes principais que possuem respectivamente 47,67% e 23,18% da variância original, e juntos explicam 70,85% da variância total.

O resultado dos K-means para 4 clusters pode ser visto na figura 13, junto com a variável *target*.

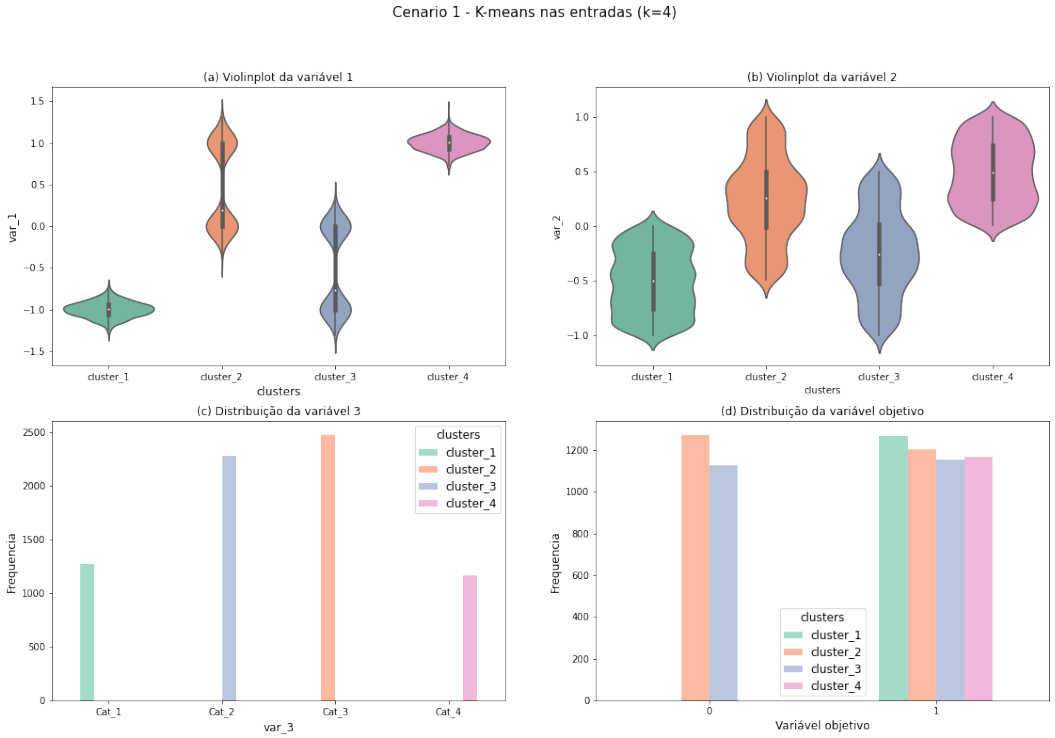
Figura 13: K-means nas entradas para k=4 no cenário 1



Fonte: Autoria própria

A partir desse resultado, podemos gerar o gráfico para cada variável por grupo, como mostrado na figura 14:

Figura 14: Distribuição das variáveis para o K-means nas entradas para k=4 no cenário 1

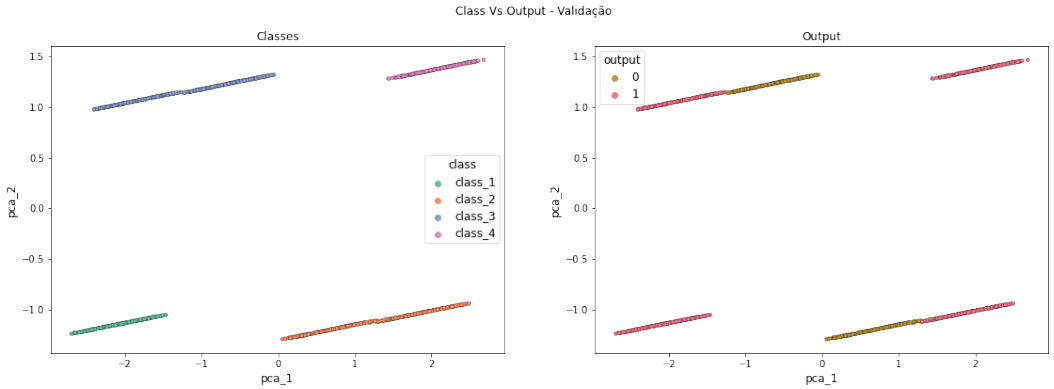


Fonte: Autoria própria

Podemos ver que cada grupo é caracterizado exclusivamente por uma categoria na variável 3, e que as demais variáveis impactam bem menos na caracterização dos clusters. Também é válido observar que os clusters 1 e 4 possuem um único *output*, mas os clusters 2 e 3 possuem os dois outputs de forma bem presente.

Para os dados de validação, o resultado seguiu o mesmo comportamento dos dados de treino. A imagem dos clusters no espaço gerado pelo PCA pode ser visto na figura 15 abaixo.

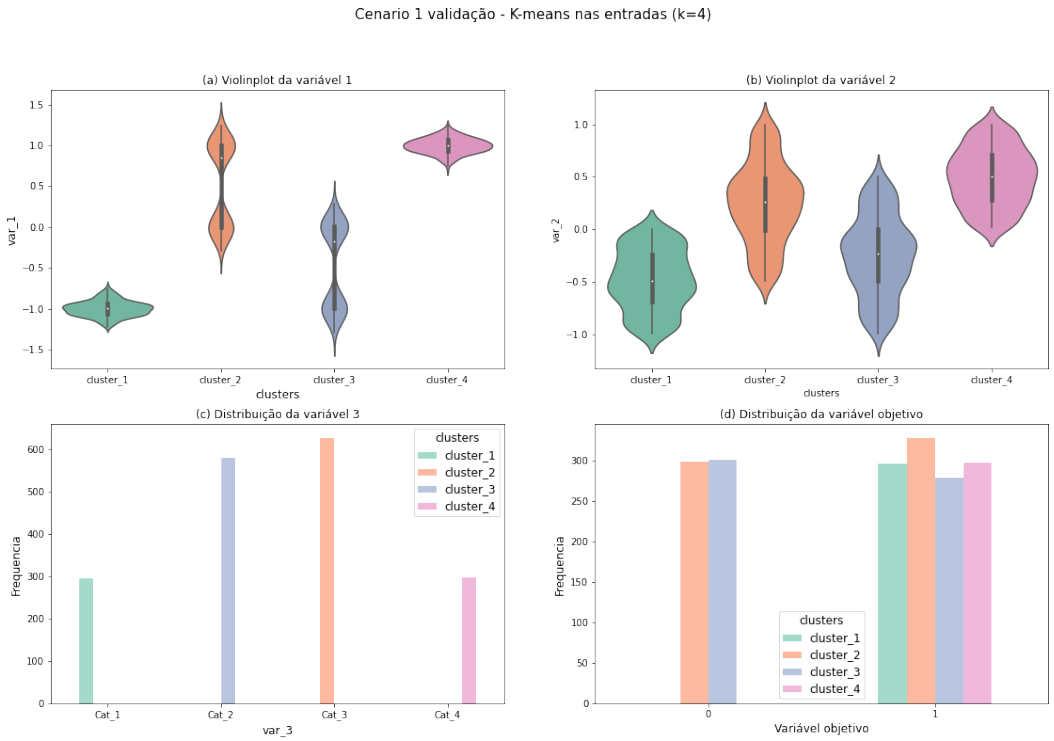
Figura 15: Validação K-means nas entradas para k=4 no cenário 1



Fonte: Autoria própria

Os demais gráficos de variáveis por cluster na validação também seguiram a distribuição dos dados de treino, como podemos observar abaixo:

Figura 16: Distribuição das variáveis em relação ao clusters no cenário 1 em validação



Fonte: Autoria própria

4.1.4.2 Analise do resultado do K-means com k=4

A analise do resultado pode ser feita de forma quantitativa, onde calulamos a pureza e geramos a tabela 1 com os seguinte dados:

Tabela 1: Pureza dos clusters K-means nas entradas k=4

Cluster	Pureza treino	Pureza validação
1	1,0	1,0
2	0,514	0,523
3	0,507	0,519
4	1,0	1,0
Total	0,676	0,679

Fonte: Autoria própria

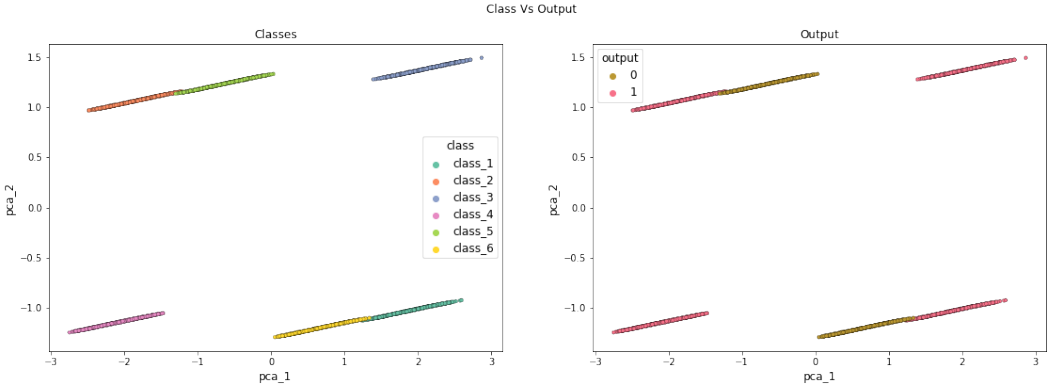
Vemos que a pureza dos clusters 1 e 4 é máxima, mas dos clusters 2 e 3 é de aproximadamente $0,5$.

Indo além, podemos fazer uma análise qualitativa e tirar algumas conclusões sobre os dados a partir dos grupos encontrados. Uma delas é que pertencer às categorias 1 e 4 implica em ter uma variável *target* 1, como mostram os clusters 1 e 4. Outra é que pertencer às categorias 2 e 3 não é informação suficiente para inferir a variável *target*.

4.1.4.3 K-means nas entradas para k=6

Forçando o K-means para 6 clusters, temos um resultado interessante. Ele pode ser visto na figura 17 abaixo, onde se mantém o espaço gerado pelo PCA.

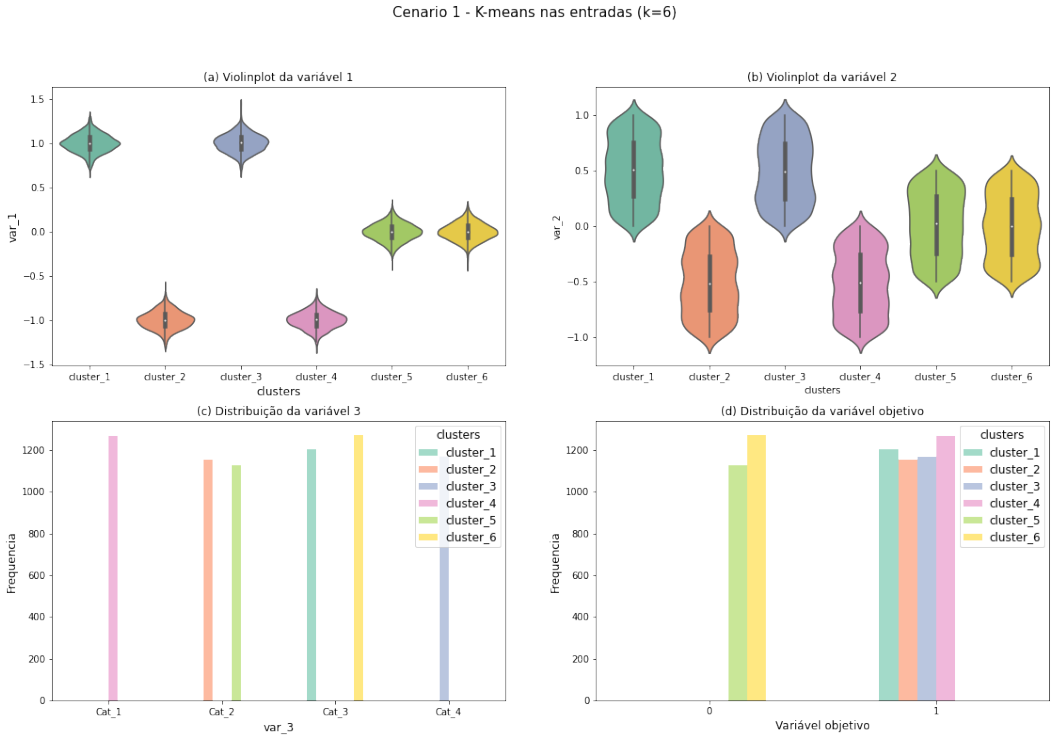
Figura 17: K-means nas entradas para k=6 no cenário 1



Fonte: Autoria própria

Podemos também gerar o gráfico para cada variável por grupo, como mostrado na figura abaixo:

Figura 18: Distribuição das variáveis para o K-means nas entradas para k=6 no cenário 1

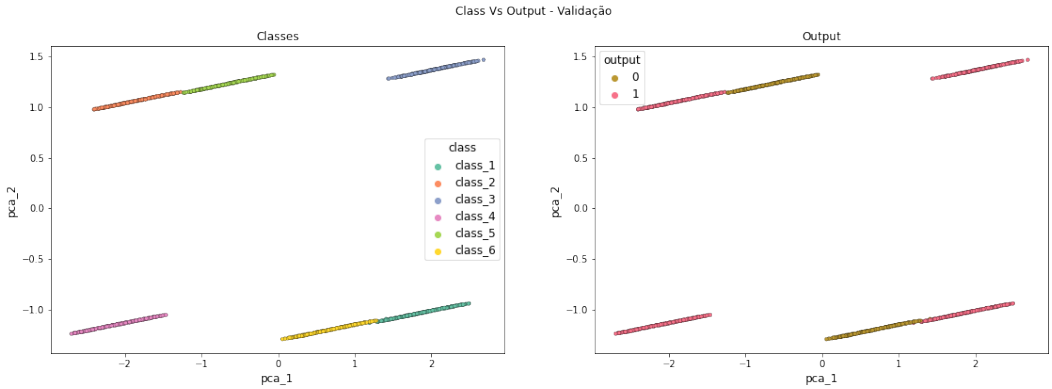


Fonte: Autoria própria

Podemos ver que, ao forçar k=6, cada grupo possui apenas uma categoria da variável 3, mas, diferente do k=4, agora alguns grupos não são caractetizados exclusivamente por sua categoria. É o caso dos clusters 2 e 5, que se diferenciam pela variável 1, e dos clusters 1 e 6, que também se deferenciam pela variável 1.

Para os dados de validação, os resultados foram iguais aos dados de treino. A imagem dos clusters no espaço gerado pelo PCA pode ser visto na figura 19 abaixo.

Figura 19: Validação K-means nas entradas para k=6 no cenário 1



Fonte: Autoria própria

Os demais gráficos por variável obtiveram o mesmo resultado que os dados de treino 18, por isso não vamos expô-los.

4.1.4.4 Analise do resultado do K-means nas entradas para k=6

Da mesma forma que a análise anterior, vamos calcular a pureza do resultado obtido:

Tabela 2: Pureza dos clusters K-means nas entradas k=6

Cluster	Pureza treino	Pureza validação
1	1,0	1,0
2	1,0	1,0
3	1,0	1,0
4	1,0	1,0
5	1,0	1,0
6	1,0	1,0
Total	1,0	1,0

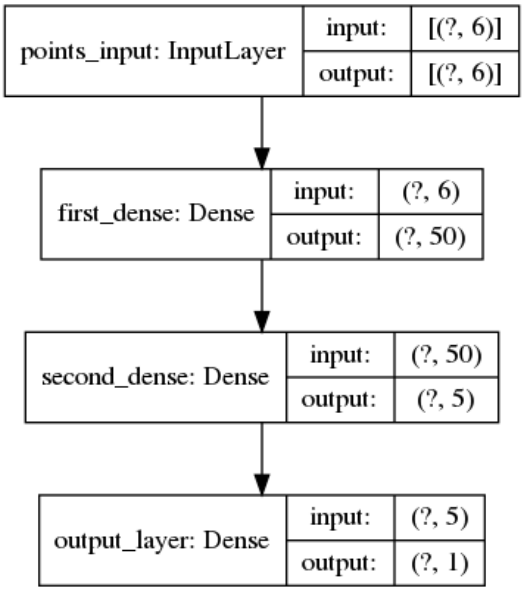
Fonte: Autoria própria

Pela análise qualitativa, concluimos que pertencer às categorias 1 e 4 implica em ter uma variável *target* 1, da mesma forma que o resultado anterior. Porém, diferente de antes, a categoria 2 possui os grupos 2 e 5 que são separados na variável 1 pela região $[-1,5;-0,5]$ e $[-0,5;0,5]$, respectivamente. Da mesma forma, os grupos 1 e 6, que estão na categoria 3, são separados na variável 1 pela região $[0,5;1,5]$ e $[-0,5;0,5]$, respectivamente. Assim, conseguimos descrever os dados de forma a segmentar a variável *target* completamente em todos os casos.

4.1.5 K-means no espaço de latência

O primeiro passo é treinar uma rede neural. Para isso, utilizamos uma rede de arquitetura de acordo com a figura 20 e com mais detalhes no código 3. Nela, a camada de entrada recebe 6 entradas, a primeira camada escondida é uma *Dense* com 50 neurônios e com função de ativação *relu*, seguida de uma segunda camada escondida *Dense* com 5 neurônios e com a mesma função de ativação *relu*, e por último, uma camada de saída com uma *Dense* de 1 neurônio e com função de ativação sigmoide.

Figura 20: Arquitetura da Rede Neural cenário 1



Fonte: Autoria própria

Código 3: Arquitetura da Rede cenário 1

```
model_input = Input(shape=(6,)
                    ,name='points_input')

first_dense = Dense(50, activation='relu',
            kernel_initializer='he_uniform',
            name='first_dense')(model_input)
second_dense = Dense(5, activation='relu',
            kernel_initializer='he_uniform',
            name='second_dense')(first_dense)
target = Dense(1, activation='sigmoid',
            name='output_layer')(second_dense)

model = Model(inputs=model_input,
```

```
14                outputs=target)
```

Fonte: Autoria própria

Como parâmetros de treinamento, utilizamos o *'adam'*, a função de custo *'binary_crossentropy'* e a métrica de *'accuracy'*, como mostrado no código 4 abaixo.

Código 4: Parâmetros de treinamento cenário 1

```
1 model.compile(optimizer='adam',
2                loss='binary_crossentropy',
3                metrics=['accuracy'])
```

Fonte: Autoria própria

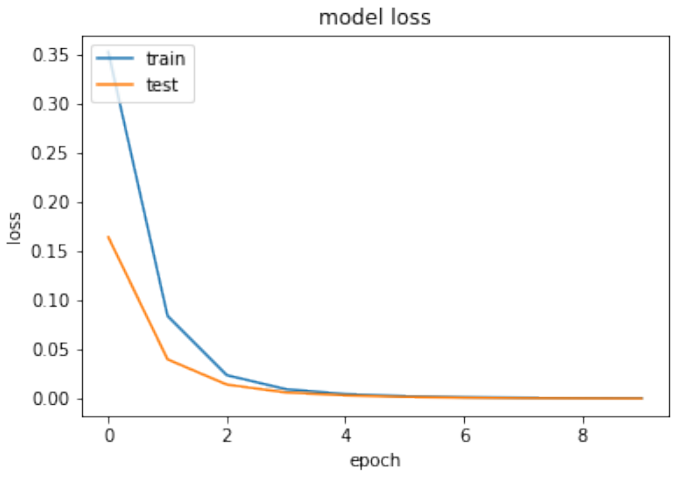
Também utilizamos parte dos dados de treino para teste da acurácia e perda do modelo, além de um *batch_size* de 10 amostras. Treinamos por 10 épocas, como pode ser visto no código 5. Os gráficos de perda e acurácia ao longo do treinamento pode ser visto nas figuras 21 e 22, respectivamente.

Código 5: Treinamento Rede cenário 1

```
1 history = model.fit(X_train, y_train,
2                validation_data=(X_test, y_test),
3                verbose = 0,
4                batch_size = 10,
5                epochs = 10)
```

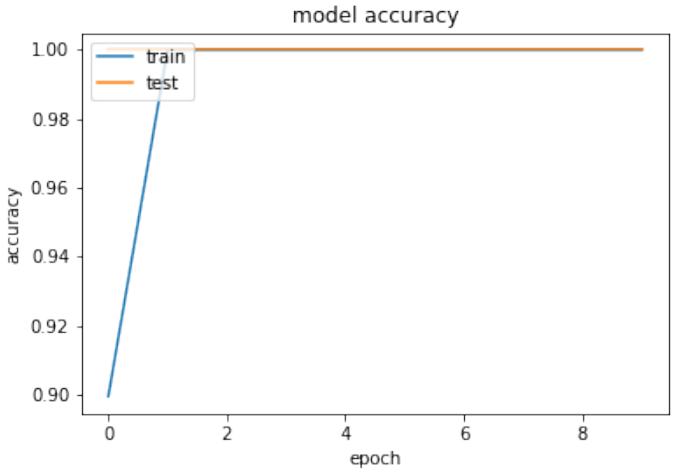
Fonte: Autoria própria

Figura 21: Perda do modelo por época de treino cenário 1



Fonte: Autoria própria

Figura 22: Acurácia do modelo por época de treino cenário 1

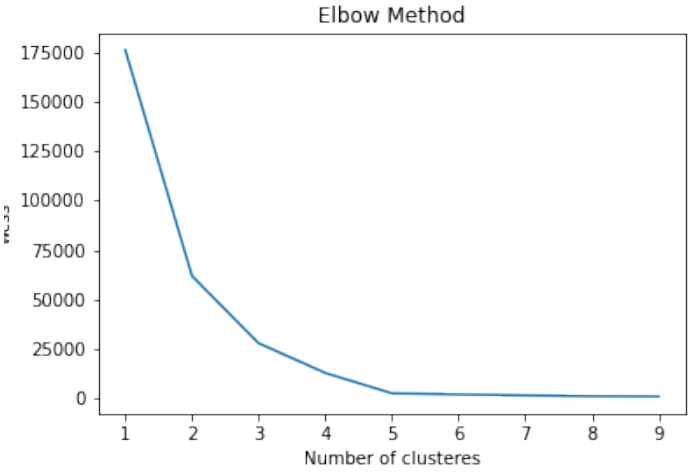


Fonte: Autoria própria

Podemos ver que o modelo teve a acurácia máxima de 100% de acerto, tanto nos dados de treino quanto nos de teste. Uma vez que temos o modelo treinado, podemos gerar os *feature vectors* para os dados. Isso é feito segmentando a rede neural e utilizando a última camada escondida como nova saída.

Ao aplicar o K-means diretamente nos *feature vectors*, o primeiro parâmetro a ser definido é o número de clusters. Para tal definição, o *Elbow Method* auxilia. Ao gerar o gráfico da figura 23 abaixo, vemos que 3 é o número de clusters ideal.

Figura 23: *Elbow Method* no espaço de latência no cenário 1



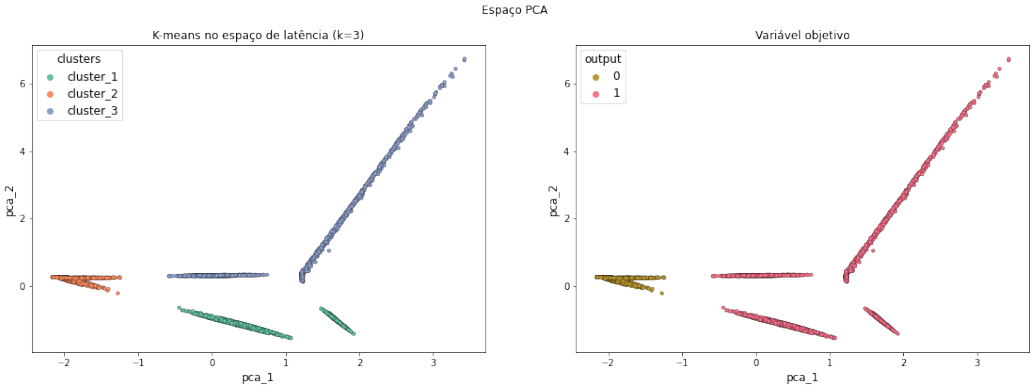
Fonte: Autoria própria

4.1.5.1 K-means no espaço de latência para k=3

Para visualizar os clusters, o método PCA será utilizado. Aplicando este método nos *feature vectors* normalizados, encontramos 2 componentes principais que possuem respectivamente 56,33% e 29,81% da variância original, e juntos explicam 86,14% da variância total.

O resultado dos K-means para 3 clusters pode ser visto na figura 24 abaixo no espaço gerado pelo PCA.

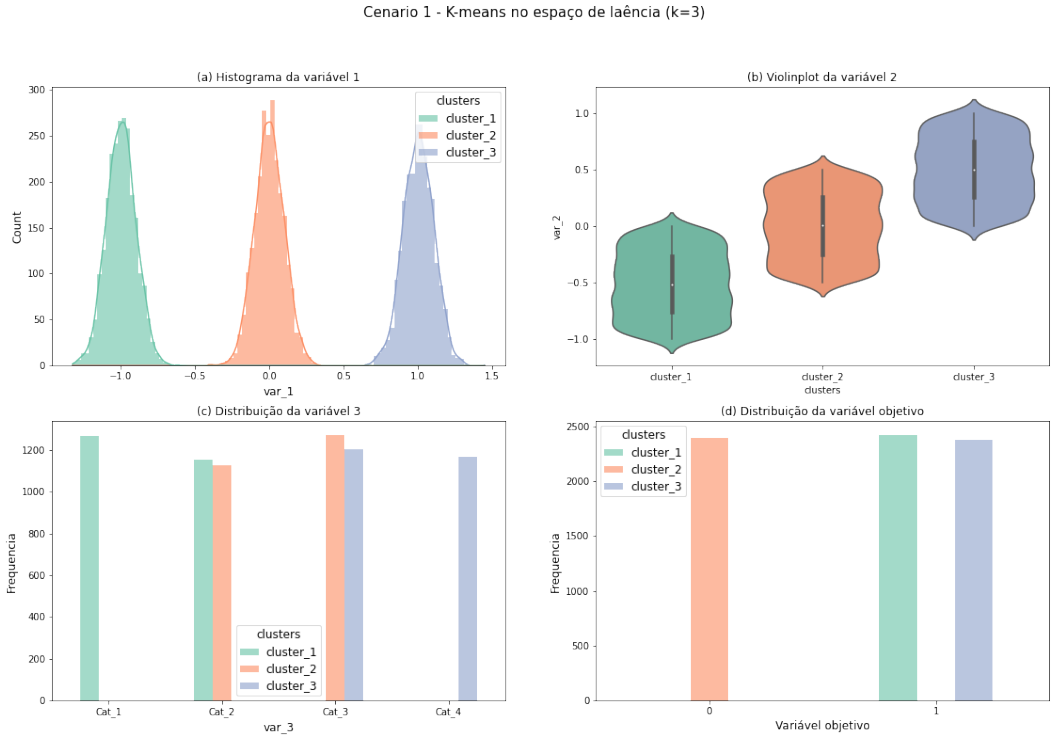
Figura 24: K-means no espaço de latência cenário no 1



Fonte: Autoria própria

A partir desse resultado, podemos gerar o gráfico para cada variável por grupo, como mostrado na figura 25:

Figura 25: Distribuição das variáveis para o K-means no espaço de latência para k=3 no cenário 1

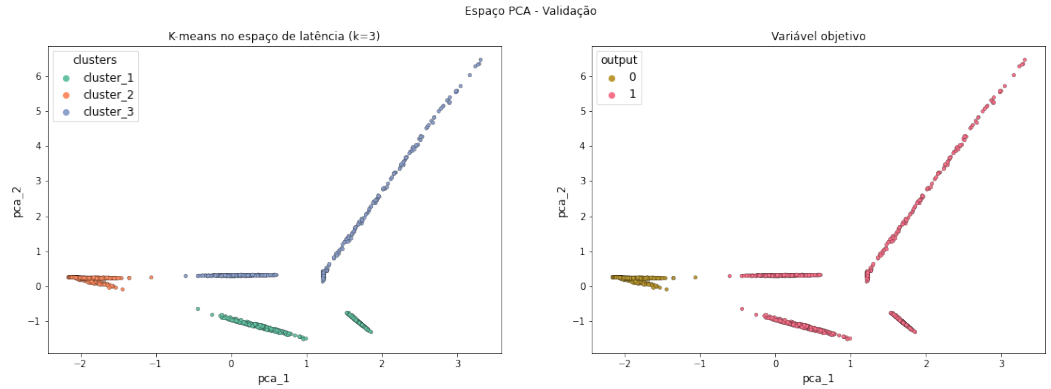


Fonte: Autoria própria

Podemos ver que cada grupo pode ser caracterizado exclusivamente pela variável 1, e que as demais variáveis não apresentam tanto impacto nos clusters. Também é válido observar que cada cluster possui um único output.

Para os dados de validação, os resultados foram iguais aos dados de treino. A imagem dos clusters no espaço gerado pelo PCA pode ser visto na figura 26 abaixo.

Figura 26: Validação K-means no espaço de latência no cenário 1



Fonte: Autoria própria

Os demais gráficos por variável obtiveram o mesmo resultado que os dados de treino na figura 25, por isso não vamos expô-los.

4.1.5.2 Análise do resultado do K-means com k=3

Calculando a pureza, obtemos a seguinte tabela:

Tabela 3: Pureza dos clusters K-means no espaço de latência para k=3

Cluster	Pureza treino	Pureza validação
1	1,0	1,0
2	1,0	1,0
3	1,0	1,0
Total	1,0	1,0

Fonte: Autoria própria

Vemos que cada grupo tem pureza máxima, e que a pureza total também é máxima. Isso significa que cada cluster encontrado possui exclusivamente um único valor da variável *target*.

Analisando os atributos dos clusters, concluímos que a variável 1 é suficiente para diferenciar os dados. A partir dela, encontramos três seções, $[-1,5; -0,5]$, $[-0,5; -0,5]$ e $[0,5; 1,5]$, cada uma com um único valor da variável *target*.

4.1.6 Comparação entre modelos

Fazendo uma comparação quantitativa, observamos na tabela 4 que a abordagem K-means nas entradas só conseguiu uma pureza máxima quando forçamos o número de clusters para 6. Por outro lado, quando aplicamos o K-means no espaço de latência, encontramos a pureza máxima com apenas 3 clusters.

Tabela 4: Pureza dos clusters por modelos no cenário 1

Modelo	Pureza treino	Pureza validação
K-means (k=4) nas entradas	0,676	0,679
K-means (k=6) nas entradas	1,0	1,0
K-means (k=3) no espaço de latência	1,0	1,0

Fonte: Autoria própria

Neste cenário fica evidente que o K-means aplicado diretamente nas entradas fica enviesado a particionar os dados primeiro pelas variáveis categórias e só depois pelas numéricas. Ainda, o resultado encontrado por esse método não é satisfatório uma vez que não ajuda a entender o conjunto de dados no cenário específico, apenas mostra combinações entre as variáveis.

Por outro lado, a técnica de aplicar o K-means no espaço de latência gerado por uma rede neural treinada apresentou um resultado mais que satisfatório, pois permitiu entender o conjunto de dados específico para o problema ao separar os grupos pela variável 1. Esse resultado se iguala às regras utilizadas para a criação do *dataset*.

4.2 Cenário 2

Este próximo cenário tem por objetivo mostrar o funcionamento da técnica de clusterização no espaço de latência em um conjunto de dados simples, mas não completamente separável. Ou seja, vamos aumentar a complexidade do cenário anterior.

Para isso, será utilizado um outro conjunto de dados gerados artificialmente, formado a partir de novas regras. Assim como no cenário 1, seguiremos as seguintes etapas: apresentar os dados gerados; análisar o conjunto de dados; preparar os dados para a modelagem; treinar modelos de K-means diretamente nas entradas e no espaço de latência com exposição dos resultados; e comparar as abordagens.

4.2.1 Dados gerados

Os dados gerados seguem o mesmo formato que o *dataset* do cenário 1, ou seja, possuem 3 variáveis de entrada e uma de saída. Essas variáveis são geradas a partir do mesmo código 1, em que cada variável segue uma distribuição definida por parâmetros.

Para este problema, utilizamos 3 regras, sendo cada uma definida como:

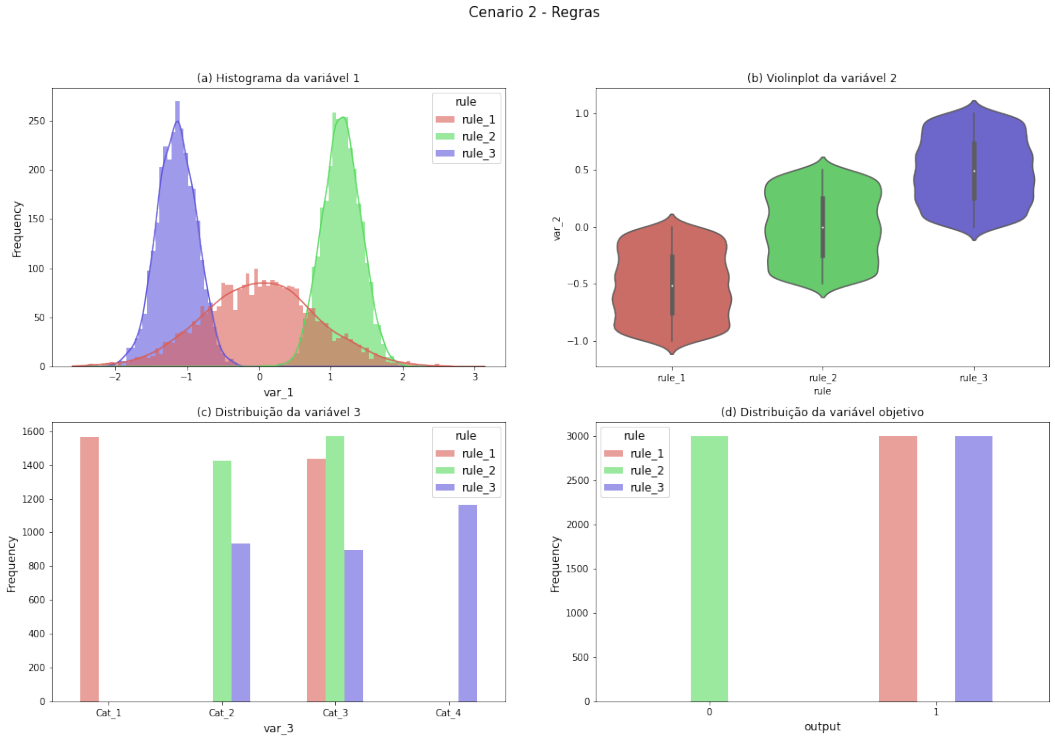
Código 6: Regras cenário 2

```
rule_1 = generate_data(
    data_name = 'rule_1',
    data_size = 3000,
    param_1 = {'mu':0, 'sigma':0.8},
    param_2 =  {'inf_lim':-1, 'sup_lim':0},
    param_3 = {'choices':['Cat_1', 'Cat_3'], 'weights':[0.5, 0.5]},
    out_param = 1
)

rule_2 = generate_data(
    data_name = 'rule_2',
    data_size = 3000,
    param_1 = {'mu':1.15, 'sigma':0.27},
    param_2 = {'inf_lim':-0.5, 'sup_lim':0.5},
    param_3 = {'choices':['Cat_2', 'Cat_3'], 'weights':[0.5, 0.5]},
    out_param = 0
)

rule_3 = generate_data(
    data_name = 'rule_3',
    data_size = 3000,
    param_1 = {'mu':-1.15, 'sigma':0.27},
    param_2 = {'inf_lim':0, 'sup_lim':1},
    param_3 = {'choices':['Cat_2', 'Cat_3','Cat_4'], 'weights':[0.3,
   0.3, 0.4]},
    out_param = 1
)
```

Fonte: Autoria própria

Os dados de gerados são mostrados a seguir na figura 27, separados por variável e por regra.

Figura 27: Variáveis geradas no cenário 2 por regra



Fonte: Autoria própria

De acordo com o histograma (a), fica visível que a variável 1 é bem característica de cada regra, mas há intersecções entre as regras.

Já no *violin plot* (b), da variável 2, vemos uma intersecção entre a regra 2 com as demais regras. Porém, nos extremos inferior, abaixo de $-0,5$ e superior, acima de $0,5$, estão presente apenas as regras 1 e 3, respectivamente.

A variável 3 é uma variável categórica com quatro categorias. Pela figura (c), a categoria 1 está presente apenas na regra 1, e a categoria 4 é encontrada apenas na regra 3. Na categoria 2 são observadas as regras 2 e 3 e, por fim, a categoria 3 aparece em todas as regras.

A variável final de output tem valor 1 para as regras 1 e 3 e valor 0 para a regra 2.

A partir dessas informações, vemos que existe um subconjunto neste *dataset* em que não é possível dizer com acurácia de 100% qual é a saída esperada apenas com as entradas. Este subconjunto é formado pela intercecção das regras 1 e 2 e pode ser identificado pelas condições seguintes:

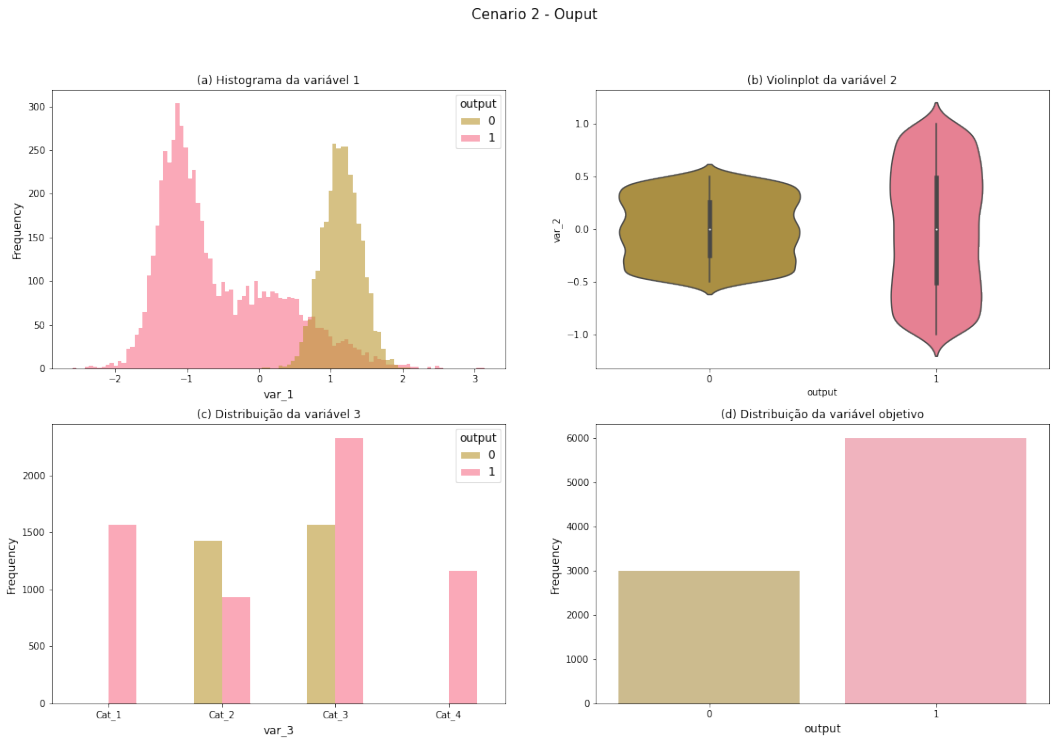
- Variável 1 entre $[0,091; 2,0619]$
- Variável 2 entre $[-0,5; 0,0]$
- Variável 3 pertencendo à categoria 3

Por conta disso, dizemos o conjunto de dados é não separável.

4.2.2 Exploração dos dados

Ao olhar o conjunto de dados sem as regras, apenas com as variáveis geradas, temos a figura 28.

Figura 28: Distribuição das variáveis em relação ao output no cenário 2



Fonte: Autoria própria

De acordo com o histograma (a), fica visível que há intersecções entre output onde o valor da variável 1 é maior que $0,09$ e menor que $2,06$.

No *violin plot* (b), nos extremos inferior, abaixo de $-0,5$, e superior, acima de $0,5$, o *output* é bem definido, mas é misto para o restante da variável 2.

De acordo com o gráfico de barras (c), vemos que o *output* é único para as categorias 1 e 4, mas é misto para as categorias 2 e 3.

Pela distribuição (d), conclui-se que o conjunto de dados é desbalanceado, uma vez que a variável de saída 0 aparece bem menos que a 1.

4.2.3 Preparação dos dados

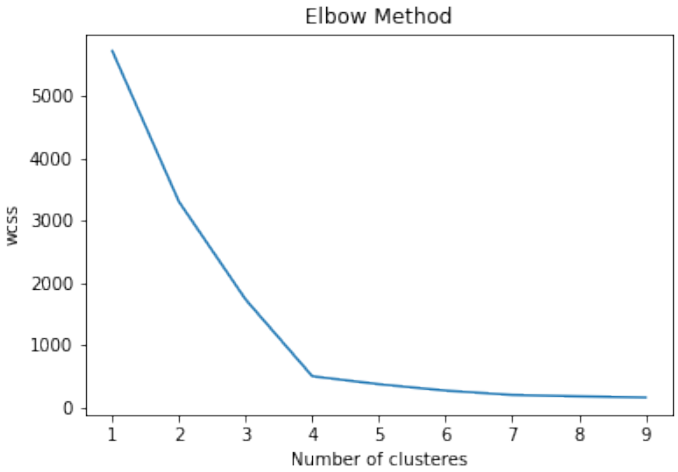
Como os dados de entrada são do mesmo formato do cenário 1, a preparação dos dados será a mesma. O conjunto será dividido em 2 subconjuntos, um para treino e outro de validação. Os subconjuntos de treino e validação serão os mesmos utilizados no modelo K-means diretamente nas entradas e K-means no espaço de latência.

Da mesma forma, as colunas numéricas variável 1 e variável 2 serão normalizadas, e na coluna categórica variável 3 será aplicada a codificação one-hot.

4.2.4 K-means nas entradas

Para definir o número de clusters, vamos utilizar o *Elbow Method*. De acordo com o gráfico 29, vemos que 4 é o número de clusters ideal. Para fins demonstrativos vamos novamente forçar em 6 clusters.

Figura 29: *Elbow Method* nas entradas no cenário 2



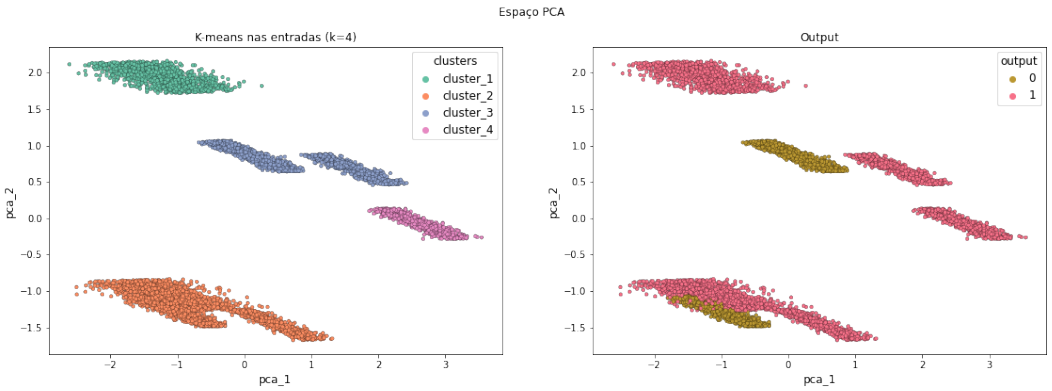
Fonte: Autoria própria

4.2.4.1 K-means com k=4

Aplicando o PCA nas entradas para a visualização, encontramos 2 componentes principais que possuem respectivamente 33,53% e 25,23% da variância original, e juntos explicam 58,77% da variância total. É importante ressaltar que a variância total representada é baixa e que, portanto, a visualização do espaço gerado fica prejudicada ao não representar grande parte da variância.

O resultado dos K-means para 4 clusters pode ser visto na figura 30 abaixo, juntamente com a variável *target*.

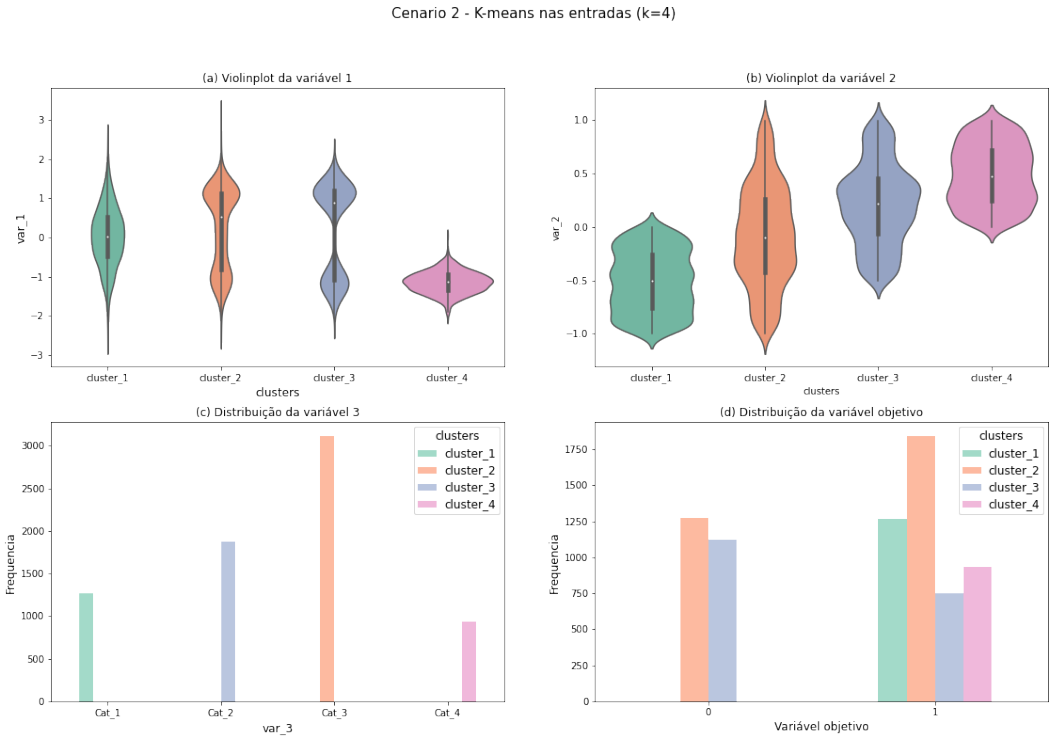
Figura 30: K-means nas entradas para k=4 no cenário 2



Fonte: Autoria própria

A partir desse resultado, podemos gerar o gráfico para cada variável por grupo, como mostrado na figura 31:

Figura 31: Distribuição das variáveis para o K-means com k=4 no cenário 2

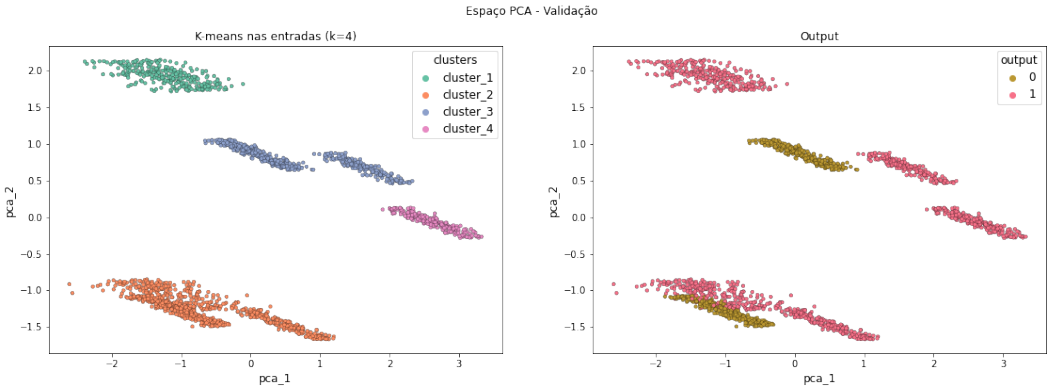


Fonte: Autoria própria

Podemos ver que cada grupo é caracterizado exclusivamente por uma categoria, e que as demais variáveis de entrada não impactam nos clusters. Também é válido observar que os clusters 1 e 4 possuem apenas um único valor da variável *target*, mas os clusters 2 e 3 possuem os dois valores de forma bem presente.

Para os dados de validação, os resultados foram iguais aos dados de treino. A imagem dos clusters no espaço gerado pelo PCA pode ser visto na figura 32 abaixo.

Figura 32: Validação K-means nas entradas para k=4 no cenário 2



Fonte: Autoria própria

Os demais gráficos por variável obtiveram o mesmo resultado que os dados de treino na figura 31, por isso não vamos expô-los.

4.2.4.2 Análise do resultado do K-means nas entradas para k=4

A analise do resultado pode ser feita de forma quantitativa, no qual calulamos a pureza e geramos a tabela 6 com os seguinte dados:

Tabela 5: Pureza dos clusters K-means nas entradas para k=4

Cluster	Pureza treino	Pureza validação
1	1,0	1,0
2	0,591	0,620
3	0,600	0,619
4	1,0	1,0
Total	0,719	0,731

Fonte: Autoria própria

Vemos que a pureza dos clusters 1 e 4 é máxima, mas dos clusters 2 e 3 é de aproximadamente $0,6$. A pureza total foca em torno de $0,7$.

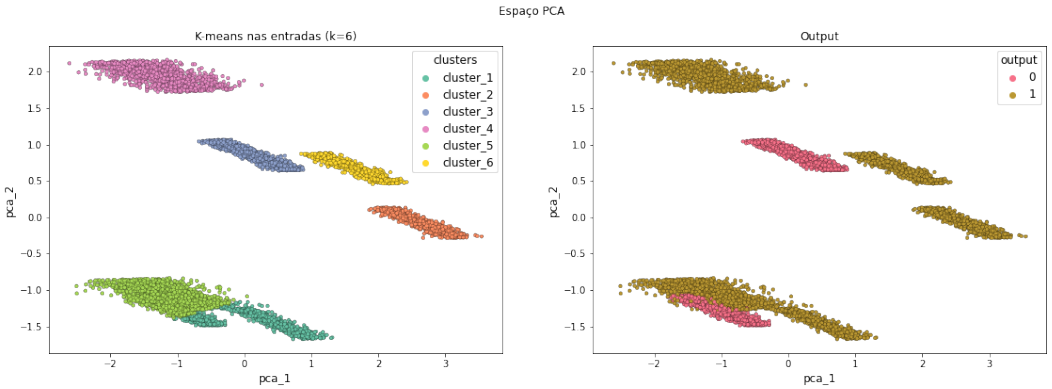
Indo além, podemos fazer uma análise qualitativa e tirar várias conclusões sobre os dados a partir dos grupos encontrados. Uma delas é que pertencer às categorias 1 e 4 implica em ter uma variável *target* 1, como mostram os clusters 1 e 4. Outra é que pertencer às categorias 2 e 3 não é informação suficiente para inferir a variável *target*.

Como o K-means particionou apenas pelas categorias, não conseguimos inferir nada das outras variáveis de entrada que nos ajude a entender o cojunto de dados e o cenário.

4.2.4.3 K-means com k=6

Forçando o K-means para 6 clusters, temos um resultado interessante. Ele pode ser visto na figura 33 abaixo, onde se mantém o espaço gerado pelo PCA.

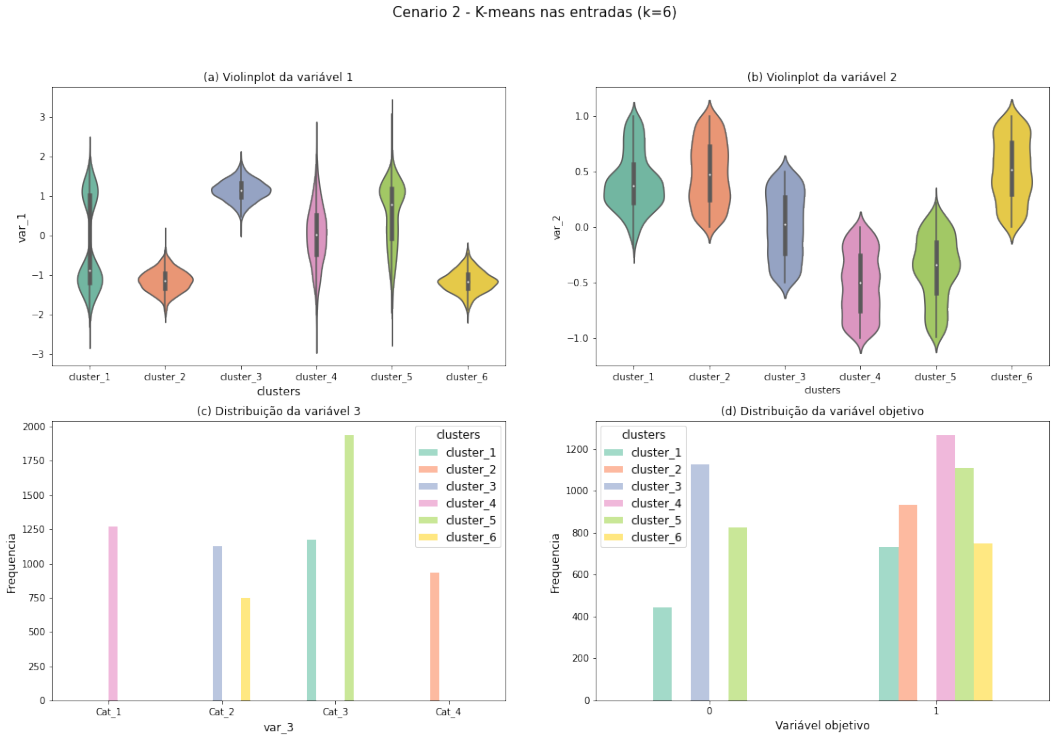
Figura 33: K-means nas entradas para k=6 no cenário 2



Fonte: Autoria própria

A partir desse resultado, podemos gerar o gráfico para cada variável por grupo, como mostrado na figura 34:

Figura 34: Distribuição das variáveis para o K-means nas entradas para k=6 no cenário 2

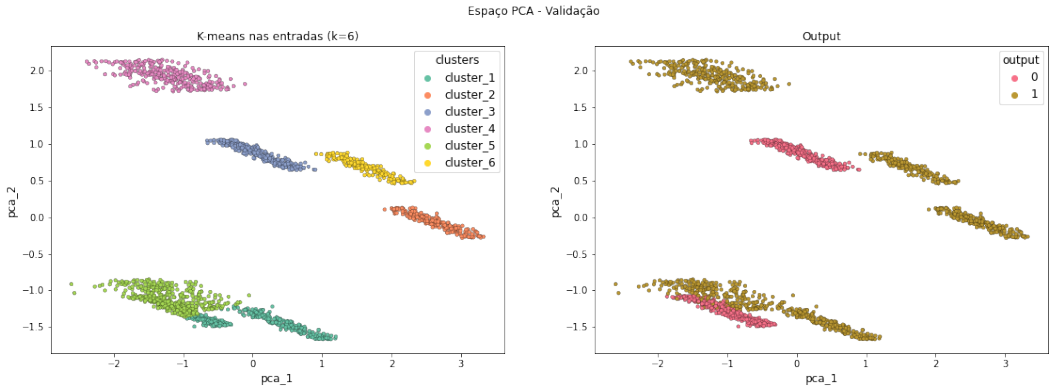


Fonte: Autoria própria

O resultado agora é que cada cluster tem apenas uma categoria, mas os clusters 3 e 6, que compartilham a categoria 2, são diferenciados pela variável 1, em que o cluster 3 é composto pela variável 1 maior que 0 e o cluster 6 pela variável 1 menor que 0. Os clusters 1 e 5, que compartilham a categoria 3, são diferenciados tanto pela variável 1 quanto pela variável 2.

Para os dados de validação, os resultados foram iguais aos dados de treino. A imagem dos clusters no espaço gerado pelo PCA pode ser visto na figura 35 abaixo.

Figura 35: Validação K-means nas entradas para k=6 no cenário 2



Fonte: Autoria própria

Os demais gráficos por variável obtiveram o mesmo resultado que os dados de treino na figura 34, por isso não vamos expô-los.

4.2.4.4 Análise do resultado do K-means nas entradas para k=6

Da mesma forma que a análise anterior, vamos calcular a pureza do resultado obtido:

Tabela 6: Pureza dos clusters K-means nas entradas para k=4

Cluster	Pureza treino	Pureza validação
1	0,622	0,672
2	1,0	1,0
3	1,0	1,0
4	1,0	1,0
5	0,57	0,583
6	1,0	1,0
Total	0,823	0,833

Fonte: Autoria própria

Vemos que a pureza dos clusters 2, 3, 4 e 6 é máxima, mas a dos clusters 1 e 5 não.

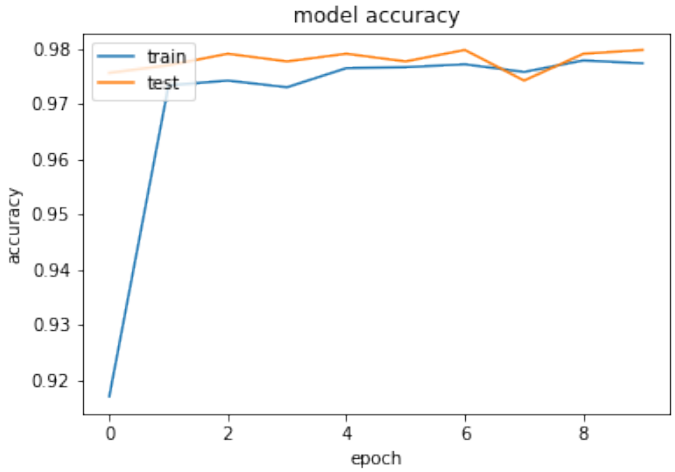
Analisando os atributos, vemos que cada cluster possui uma única categoria. Pertencer às categorias 1 e 4 implica em ter uma variável *target* 1, da mesma forma que o resultado obtido em k=4. Porém, diferente de antes, a categoria 2 possui os grupos 3 e 6, que são separados na variável 1 pela região $[0,0;2,0]$ e $[-2,0;0,0]$, e possuem uma saída da variável *target* única de 0 e 1, respectivamente. Por outro lado, os clusters 1 e 5, que

compartilham a categoria 3, são separados na variável 1 e na 2, mas essa separação não é muito clara uma vez que forçamos 6 clusters totais, além de ambas possuirem saídas 0 e 1 na variável *target*.

4.2.5 K-means no espaço de latência

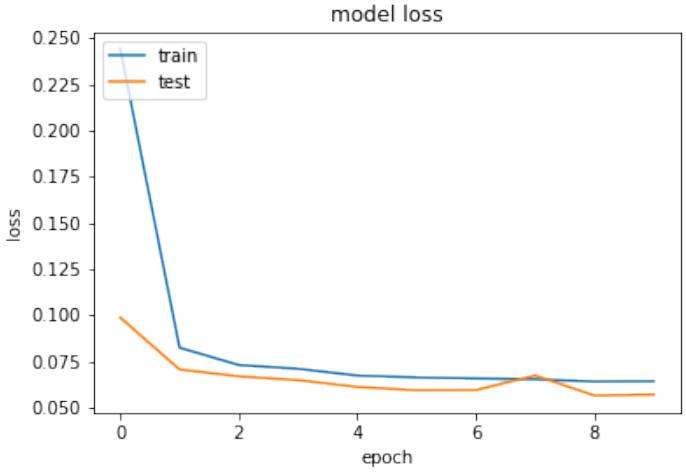
O primeiro passo é treinar uma rede neural. Para isso, utilizamos uma rede com a mesma arquitetura e hiperparâmetros descritos na seção 4.1.5. Os gráficos o treinamento com a acurácia e a perda ao longo das épocas pode ser visto nas figuras 36 e 37, respectivamentes.

Figura 36: Acurácia do modelo por época de treino Cenário 2



Fonte: Autoria própria

Figura 37: Perda do modelo por época de treino Cenário 2

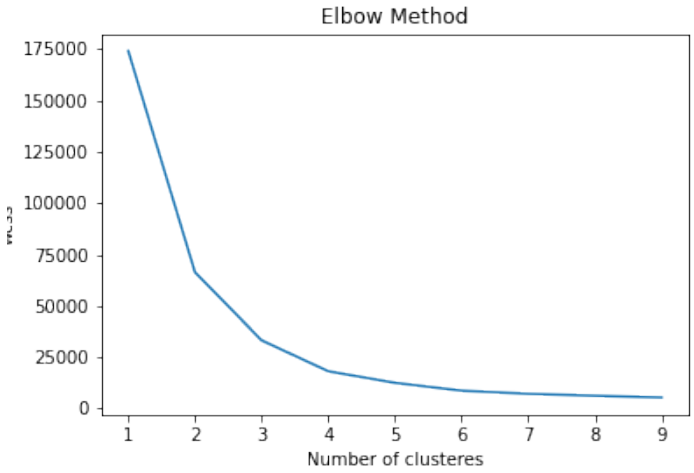


Fonte: Autoria própria

Podemos ver que o modelo teve a acurácia alta, de cerca de 98% de acerto, tanto nos dados de treino quanto nos de teste. Uma vez que temos o modelo treinado, podemos gerar os *feature vectors* para os dados.

Ao aplicar o K-means diretamente nos *feature vectors*, o primeiro parâmetro a ser definido é o número de clusters. Para tal definição, o *Elbow Method* auxilia. Ao gerar o gráfico da figura 38 abaixo, vemos que 4 é o número de clusters ideal. Para fins de demonstração, vamos também forçar 5 clusters.

Figura 38: *Elbow Method* no espaço de latência no cenário 2



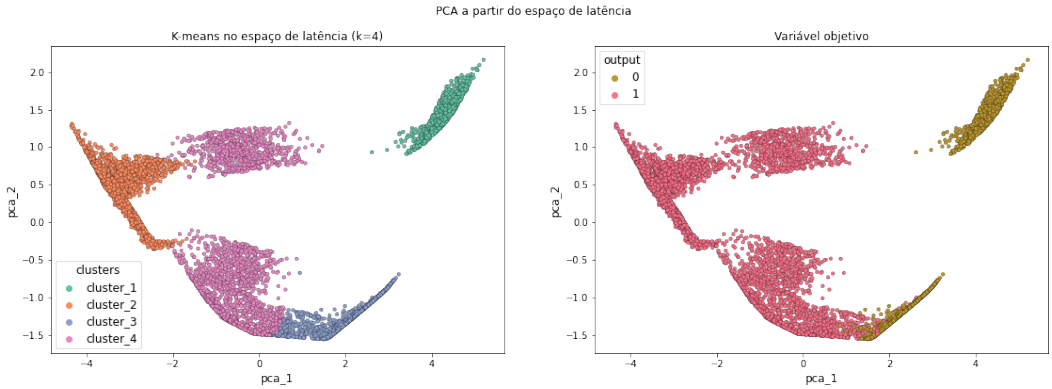
Fonte: Autoria própria

4.2.5.1 K-means no espaço de latência para k=4

Para visualizar os clusters, o método PCA será utilizado novamente. Aplicando este método nos *feature vectors*, encontramos 2 componentes principais que possuem respectivamente 79,27% e 12,20% da variância original, e juntos explicam 91,48% da variância total.

O resultado do K-means para 4 clusters pode ser visto na figura 39 abaixo no espaço gerado pelo PCA.

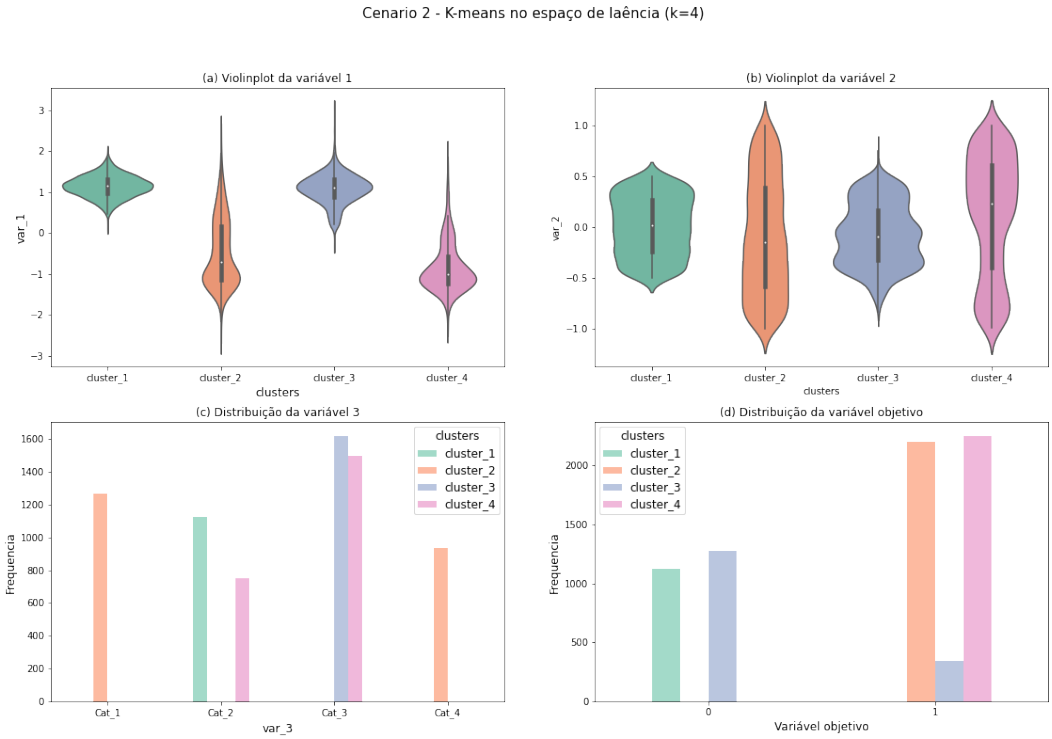
Figura 39: K-means no espaço de latência para k=4 no cenário 2



Fonte: Autoria própria

A partir desse resultado, podemos gerar o gráfico para cada variável por grupo, como mostrado na figura 40:

Figura 40: Distribuição das variáveis para o K-means no espaço de latência para k=4 no cenário 2



Fonte: Autoria própria

Podemos observar que os grupos 1, 2 e 4 possuem apenas um output, mas o grupo 3

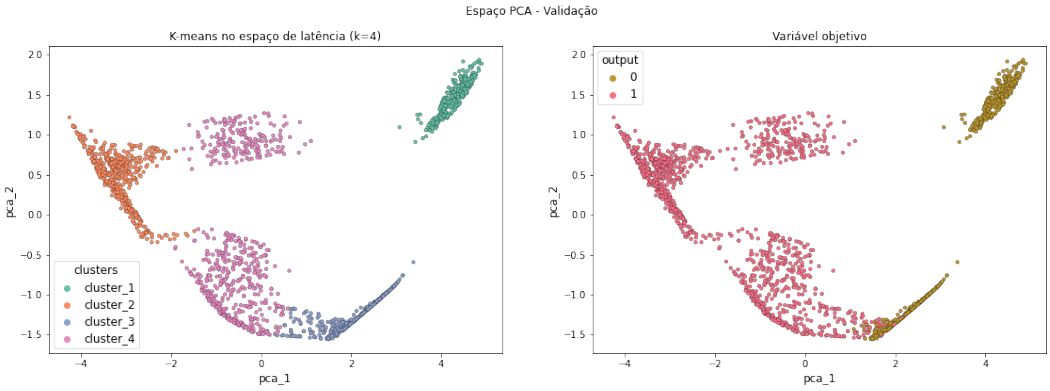
possui em grande parte output 0 e um pouco de output 1.

Ao analisarmos os grupos encontrados, podemos observar as seguintes características:

- cluster 1: possui apenas a categoria 2, com *var_1* entre $[0,0; 2,0]$. Possui exclusivamente saída 0 da variável *target*.
- cluster 2: único que possui as categorias 1 e 4, e não está presente em nenhuma outra categoria da *var_3* apenas. Possui exclusivamente saída 1 da variável *target*.
- cluster 3: possui apenas a categoria 3, *var_1* majoritariamente entre $[0,1e2,1]$ e *var_2* majoritariamente entre $[-0,6; 0,5]$. Possui principalente saída 0 da variável *target*, mas com algumas 1.
- cluster 4: possui as categorias 2 e 3, e *var_1* majoritariamente abaixo de 0.2 e *var_2* diversos, mas tendendo a valores não entre $[-0,5; 0,0]$. Possui exclusivamente saída 1 da variável *target*.

Para os dados de validação, os resultados foram iguais aos dados de treino. A imagem dos clusters no espaço gerado pelo PCA pode ser vista na figura 41 abaixo.

Figura 41: Validação K-means no espaço de latência para k=4 no cenário 2



Fonte: Autoria própria

Os demais gráficos por variável obtiveram o mesmo resultado que os dados de treino na figura 40, por isso não vamos expô-los.

4.2.5.2 Análise do resultado do K-means no espaço de latência para k=4

Calculando a pureza, obtemos a seguinte tabela:

Tabela 7: Pureza dos clusters K-means no espaço de latência para k=4

Cluster	Pureza treino	Pureza validação
1	1,0	1,0
2	1,0	1,0
3	0,786	0,781
4	1,0	1,0
Total	0,952	0,953

Fonte: Autoria própria

Conforme a tabela, vemos que apenas o grupo 3 é o único que não possui pureza máxima. Vemos também que a pureza total foi alta, 0.95.

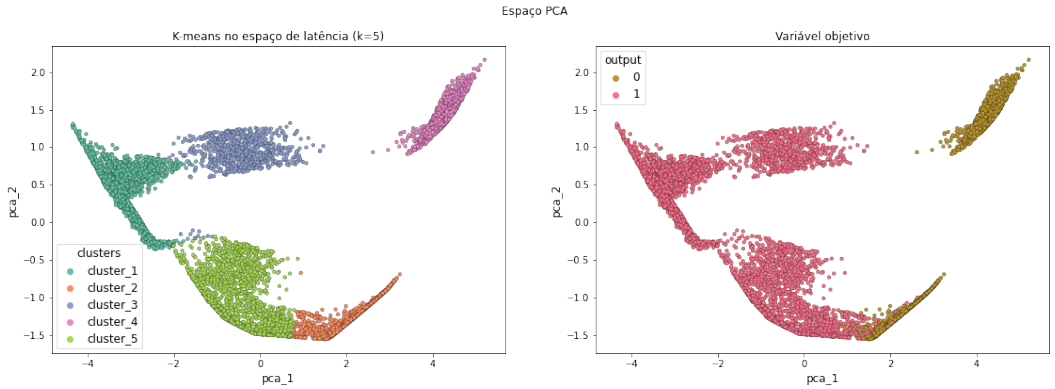
A partir dos clusters podemos identificar padrões claros no conjunto de dados. De acordo com o cluster 2 vemos que as categorias 1 e 4 implicam uma variável *target* de valor 1. Os clusters 1 e 3 se diferenciam entre si por estarem em categorias diferentes, mas possuem distribuições parecidas nas variáveis 1 e 2. Porém, o cluster 1 implica uma variável *target* de valor 0, e o cluster 3 não tem uma variável objetiva definitiva, mas possui majoritariamente 0. O cluster 4 se diferencia do 1 e do 3 pela variável 1, por estar bem mais presente na região nos valores negativos, e pela variável 2 por estar presente nos extremos.

Assim, o modelo encontrou dois perfis com *output* 1 bem definido, um com *output* 0 bem definido, um perfil com *output* misto bem definido.

4.2.5.3 K-means no espaço de latência para k=5

O resultado é o seguinte espaço:

Figura 42: K-means no espaço de latência para k=5 no cenário 2

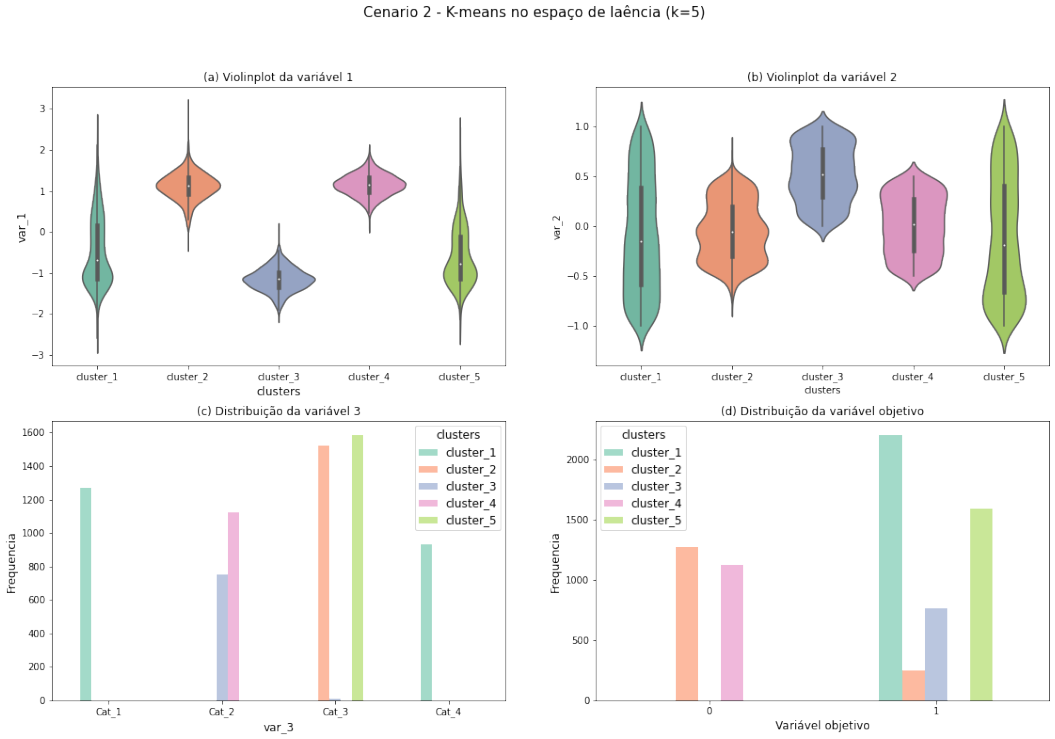


Fonte: Autoria própria

Analisando o resultado, vemos que o cluster 4 de k=4 da figura 39 foi particionado em 2.

Os gráficos por variável encontrados foram:

Figura 43: Distribuição das variáveis para o K-means no espaço de latência para k=5 no cenário 2



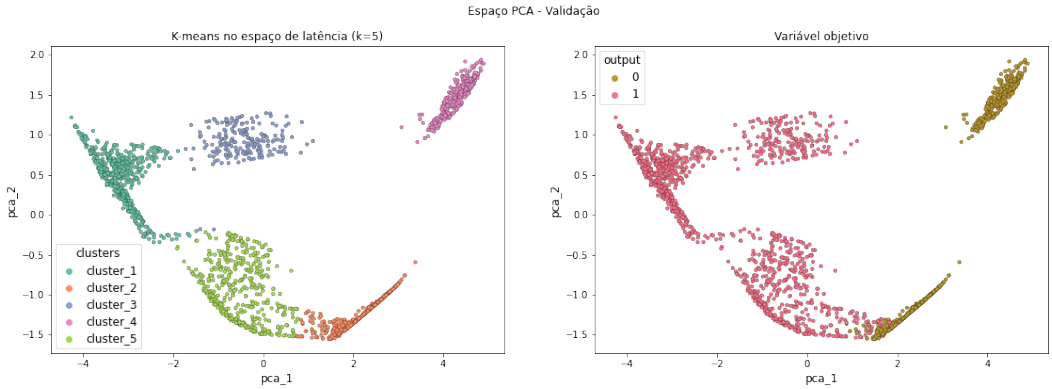
Fonte: Autoria própria

Assim, notamos as seguintes características:

- Cluster 1: antigo Cluster 2
- Cluster 2: antigo Cluster 3
- Cluster 3: categoria 2 e *var_1* menor que 0. Possui exclusivamente saída 1 da variável *target*.
- Cluster 4: antigo Cluster 1
- Cluster 5: categoria 3 apenas e *var_1* Majoritariamente abaixo de $0,2$. Possui exclusivamente saída 1 da variável *target*.

Para os dados de validação, os resultados foram iguais aos dados de treino. A imagem dos clusters no espaço gerado pelo PCA pode ser visto na figura 44 abaixo.

Figura 44: Validação K-means no espaço de latência para k=5 no cenário 2



Fonte: Autoria própria

Os demais gráficos por variável obtiveram o mesmo resultado que os dados de treino na figura 43, por isso não vamos expô-los.

4.2.5.4 Análise do resultado do K-means no espaço de latência para k=5

Calculando a pureza, obtemos a seguinte tabela:

Tabela 8: Pureza dos clusters K-means no espaço de latência para k=5

Cluster	Pureza treino	Pureza validação
1	1,0	1,0
2	0,837	0,826
3	1,0	1,0
4	1,0	1,0
5	1,0	1,0
Total	0,966	0,965

Fonte: Autoria própria

Conforme a tabela, vemos que apenas o grupo 2 não possui pureza máxima.

Em comparação ao resultado k=4 anterior, vemos que o modelo particionou ainda mais um perfil em 2. Isso resultou em um total de três perfis com output 1 bem definido, um com output 0 bem definido, um perfil misto bem definido. Porém, a quebra foi apenas por categoria e não apresentou um grande impacto no entendimento dos dados, uma vez que essa quebra não impactou na pureza dos novos clusters.

4.2.6 Comparação entre modelos

Fazendo uma comparação quantitativa, observamos pela tabela 9 que a abordagem K-means nas entradas obteve uma pureza baixa, mesmo quando forçamos o número de clusters para 6. Por outro lado, quando aplicamos o K-means no espaço de latência, encontramos uma pureza alta com apenas 4 clusters.

Tabela 9: Pureza dos clusters por modelos no cenário 2

Modelo	Pureza treino	Pureza validação
K-means (k=4) nas entradas	0,719	0,731
K-means (k=6) nas entradas	0,823	0,833
K-means (k=4) no espaço de latência	0,952	0,953
K-means (k=5) no espaço de latência	0,966	0,965

Fonte: Autoria própria

Vemos mais uma vez o K-means aplicado diretamente na entrada sendo enviesado pelas variáveis categóricas, já que a distância das categorias é mínima ou máxima caso seja da mesma categoria ou não seja, respectivamente. O resultado encontrado não foi significativo, uma vez que mostrou apenas combinações entre as categorias.

Por outro lado, a técnica de clusterizar os *feature vectors* no espaço de latência gerado pelo modelo mostrou resultados interessantes e interpretáveis. Encontramos grupos bem definidos, com *output* único, e um único grupo com output misto, praticamente o subconjunto não separável onde não é possível prever com 100% de acurácia o *output*.

4.3 Cenário 3 - Empréstimos P2P

Uma forma de *crowdfunding* criada em 2005, e que vem se popularizando nos últimos anos, é o empréstimo peer–to–peer (P2P). Nesta modalidade de crédito, pessoas se juntam para financiar um empréstimo solicitado por uma pessoa ou empresa. Cada empréstimo tem uma taxa de juros fixada de acordo com o risco da operação, e o pagamento aos credores é feito num sistema de amortização constante, o que gera uma parcela mensal que contempla o principal mais juros incidido.

O papel das *fintechs* é principalmente calcular o risco dos empréstimos e intermediar as operações. Além disso, em casos de inadimplência, efetuam a cobrança. Seu diferencial é gerar crédito de maneira mais eficiente a um custo menor que o oferecido pelos bancos. Por isso, ter um modelo de análise de risco acertivo e automatizado é essencial para atingir os objetivos da fintech: oferecer taxas menores para os tomadores de crédito e maior rendimento para investidores.

Uma das primeiras fintechs de empréstimos P2P e atualmente uma das maiores é a *Lending Club* (LC), empresa na qual baseamos este cenário.

4.3.1 *Lending Club*

Lending Club é uma empresa americana de empréstimos peer-to-peer, com sede em San Francisco, Califórnia. Foi o primeiro credor peer-to-peer a registrar suas ofertas como títulos na Securities and Exchange Commission (SEC) e a oferecer empréstimos em um mercado secundário. Em seu auge, o *Lending Club* foi a maior plataforma de empréstimo peer-to-peer do mundo.

O LC permitiu que os mutuários criassem empréstimos pessoais sem garantia entre US $ 1,000 e US $ 40,000. O período padrão do empréstimo era de três anos. Os investidores puderam pesquisar e navegar nas listas de empréstimos no site do LC e selecionar os empréstimos nos quais queriam investir com base nas informações fornecidas sobre o mutuário, montante do empréstimo, grau do empréstimo e finalidade do empréstimo. Os investidores ganharam dinheiro com os juros desses empréstimos. O *Lending Club* ganhou

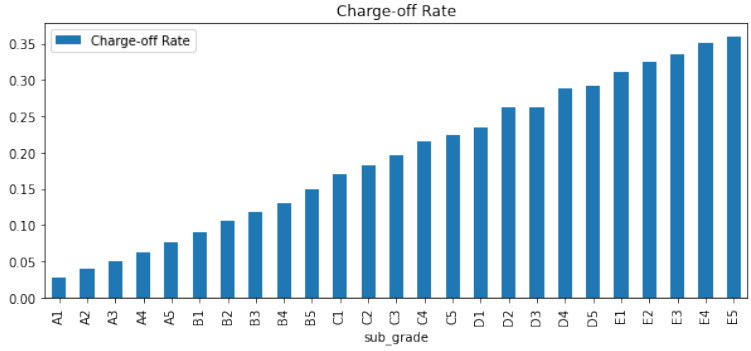
dinheiro cobrando dos mutuários uma taxa de originação e dos investidores uma taxa de serviço.

4.3.2 Entendimento do problema

A carteira do investidor é composta por um conjunto de empréstimos. Destes, parte dos devedores acabam não cumprindo sua obrigação de pagar todas as parcelas por diversos motivos, como por fraude, falência ou má administração. Assim, para o investimento ser lucrativo, o pagamento dos adimplentes devem cobrir a perda dos inadimplentes para que, na média, o rendimento total seja positivo.

Nesse contexto, a empresa mediadora deve prever qual a probabilidade de um tomador pagar ou não o empréstimo, ou seja, o risco do solicitante de empréstimo ficar inadimplente. Esse risco é associado a uma classificação chamada de *grade*. Na prática, um conjunto de empresas do mesmo *grade* deve ter uma inadimplência média controlada e, quanto pior o *grade*, maior é a inadimplência.

Figura 45: Inadimplência por *subgrades*



Fonte: Autoria própria

A figura 45 acima mostra os *grades* utilizados pela LC com suas respectivas inadimplências. Nota-se que a inadimplência progride de acordo com as *grades*. Assim, concluimos que o modelo da LC tem uma boa perfomance. Porém, com a popularização desse modelo de negócio, esse mercado fica cada vez mais competitivo e conseguir prever melhor o risco se torna uma necessidade para liderar o mercado.

Tipicamente, para realizar uma análise de risco do empréstimo, utiliza-se informações diversas, como as listadas abaixo:

Pessoa: idade, emprego, renda, se possui casa própria ou alugada, endereço, histórico de

dívidas com suas linhas de crédito, score de crédito, entre outros.

Empresa: setor de atuação, faturamento, custos, endereço, histórico de dívidas com suas linhas de crédito, score de crédito, entre outros.

Empréstimo: valor, motivo, duração, entre outros.

Visando melhorar a eficiência e a acurácia do modelo de crédito, vamos aplicar a clusterização para encontrar grupos e analisar a inadimplência de cada um.

O resultado será avalidado de forma quantitativa, comparando a inadimplência média observada em validação com a prevista pelo modelo da LC e com a dos clusters encontrados em treino.

É importante ressaltar que só temos dados de empréstimos realizados e, por isso, eles têm uma tendência a ser menos inadimplêntes que os recusados pela LC. Então só podemos desenvolver um modelo aplicável após aprovação inicial pelo modelo da LC.

4.3.3 Exploração dos dados

Para este cenário, utilizamos um conjunto de dados disponibilidaco pelo *Lending Club*, chamado de *lending-club*, e distribuídos pelo *Kaggle* [5].

Este conjunto contém empréstimos de 2015 até o último quadrimestre de 2018 e é composto de 144 colunas e 2,2 milhões de empréstimos. Ele possui informações tanto em relação ao momento anterior ao empréstimo, utilizadas para a análise de risco, quanto posteriores, referentes ao seu pagamento. Apesar de possuir diversas informações, não estão presentes neste conjunto todas as variáveis utilizadas pela LC. Por exemplo, não temos todas as variáveis utilizadas para análise de risco pela LC e falta o hitórico de pagamento das parcelas.

Das colunas disponíveis, aquelas utilizadas estão listadas na tabela 10 abaixo.

Tabela 10: Variáveis exploradas do *Lending Club*

Nome da variável	Tipo	Descrição
id	numérico	Um id único atribuido ao empréstimo
acc_open_past_24mths	numperico	O número total de linhas de crédito abertas nos ultimos 24 meses
annual_inc	numérico	Renda anual

application_type	categórico	Se o empréstimo é individual ou compartilhado
avg_cur_bal	numérico	Saldo médio atual de todas as contas
bc_open_to_buy	numérico	Total de cartões bancários rotativos disponíveis para compra
bc_util	numérico	Proporção do saldo atual e do limite de crédito
delinq_2yrs	numérico	O número de incidências de inadimplência vencidas há mais de 30 dias no histórico de crédito do mutuário nos últimos 2 anos
dti	numérico	Razão entre dívidas e renda do mutuário (*debt-to-income ratio*)
earliest_cr_line	data	Data da primeira linha de crédito
emp_length	numérico	Tempo no cargo
emp_title	categórico	Cargo no emprego
fico_range_high	inteiro	Score de crédito FICO
funded_amnt_inv	numérico	Quantia financiada
grade	categórico	Classificação de grade do empréstimo
home_ownership	categórico	Categoria de posse da casa
inq_last_6mths	numérico	Número de inquéritos nos últimos 6 meses
issue_d	data	A data em que o empréstimo foi concedido
loan_status	categórico	Situação atual do empréstimo
mort_acc	numérico	Número de hipotecas
mths_since_last_delinq	numérico	O número de meses desde a última inadimplência do mutuário
mths_since_last_major_derog	numérico	Número de meses desde o último prejuízo
mths_since_last_record	numérico	Número de meses desde o último registro público de inadimplência
num_bc_tl	numérico	Número de cartões bancários
num_il_tl	numérico	Número de empréstimos
num_sats	numérico	Número de contas satisfatórias, ou seja, contas que nunca ficaram inadimplentes por um longo tempo, tipicamente 5 anos

open_acc	numérico	O número de linhas de crédito abertas no histórico de crédito do mutuário
pct_tl_nvr_dlq	numérico	Porcentagem de linhas comerciais que nunca ficaram inadimplentes
percent_bc_gt_75	numérico	Porcentagem de todas as contas de cartão bancário maior que 75% do limite
pub_rec	numérico	Número de registros públicos de inadimplência
pub_rec_bankruptcies	numérico	Número de registros públicos de falências
purpose	categórico	Motivo do empréstimo
revol_bal	numérico	Saldo total de crédito rotativo
revol_util	numérico	Taxa de utilização da linha rotativa
sub_grade	categórico	Classifição de subgrade do empréstimo
tax_liens	numérico	Número de reclamações legais contra impostos não pagos
term	numérico	Duração de empréstimo em meses
tot_coll_amt	numérico	Montante total já devidos
tot_cur_bal	numérico	Saldo total de todas as contas
tot_hi_cred_lim	numérico	Limite de crédito total
total_acc	numérico	O número total de linhas de crédito ativas no histórico de crédito do mutuário
total_bal_ex_mort	numérico	Saldo de crédito total excluindo hipoteca
total_bc_limit	numérico	Limite de crédito total do cartão do banco
total_il_high_credit_limit	numérico	Limite de crédito de empréstimos total
total_rev_hi_lim	numérico	Limite de crédito rotativo total
verification_status	categórico	Indica se a LC verificou a renda informada
zip_code	numérico	Os primeiros 3 digitos do código postal fornecido pelo mutuário

Fonte: Autoria própria

Na exploração dos dados, analisamos cada variável individualmente. Como exemplo de descobertas, temos que a coluna *issue_d* possui a data em que o empréstimo foi concedido. A partir dela, podemos separar os dados de validação por período de tempo, e não simplesmente de maneira aleatória. Isso nos permite realizar uma validação de maneira

realista, por simular como o modelo performaria ao ser treinado em dados do começo de 2015 e colocado em produção nos últimos dois meses daquele ano.

Outra observação é que os empréstimos possuem uma duração de 36 ou 60 meses. Como a base tem início em 2015 e final em 2018, os empréstimos de maior duração não terminaram. Por isso, utilizamos apenas a parcela dos dados em que os empréstimos têm duração de 36 meses, que são grande maioria no conjunto.

Uma outra parcela dos dados que removemos foram os empréstimos conjuntos, pois eles representam uma parcela muito pequena na base e possuem outros tipos de dados dos individuos. Também removemos os empréstimos com *grade* F e G, pois também representam uma parcela muito pequena.

Algumas outras variáveis, como *loan_amt_invested*, *grade*, *subgrade* e *loan_status*, são posteriores à análise de risco e, portanto, não podem ser utilizadas como entrada do nosso modelo.

A variável *loan_status* pode ser usada como variável *target*, uma vez que ela contém o *status* atual do empréstimo. As opções podem ser *Charged Off*, caso o empréstimo seja classificado como prejuízo, *Fully Paid*, caso o empréstimo tenha sido pago por completo, ou outra categoria indicando que ele ainda não terminou. Para o nosso cenário, vamos utilizar apenas os empréstimos dados como finalizados, e a relação é de 15% inadimplente e 85% adimplentes.

Além disso, como o conjunto de dados é real, encotramos dados faltantes e outliers em algumas colunas, o que exige um tratamento.

4.3.4 Preparação dos dados

A divisão de treino e validação se deu pela data do financiamento. Os empréstimos de janeiro até outubro de 2015 foram separados para treino e os de novembro e dezembro para validação.

Da mesma forma que nos cenários anteriores, os dados categóricos foram codificados no formato one-hot. As colunas numéricas foram limitadas inferior e superiormente para evitar outliers, de acordo com a distribuição de cada uma, e depois normalizadas. Em seguida, os dados faltantes foram preenchidos por -1.

Algumas colunas foram transformadas para agregar informações mais úteis para o modelo. É o caso da coluna *emp_title* em conjunto da *annual_inc*, em que estimamos

uma faixa de renda esperada de acordo com o cargo do mutuário. Outro exemplo é o *earliest_cr_line*, do qual extraímos o ano da primeira linha de crédito.

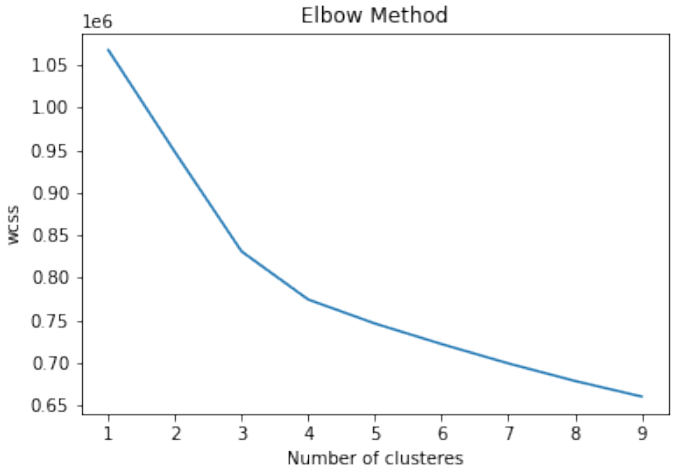
Por fim, nossa variável *target* foi formada a partir do *loan_status*. Um empréstimo classificado como *Fully Paid* (totalmente pago) recebe valor 1 e *Charged Off* (Prejuízo) valor 0.

No final da preparação, obtivemos 45 variáveis de entrada e 1 variável *target*.

4.3.5 K-means nas entradas

Para definir o número de clusters, vamos utilizar o *Elbow Method*. De acordo com o gráfico 46, vemos que 5 é um número bom de clusters.

Figura 46: *Elbow Method* nas entradas no cenário 3

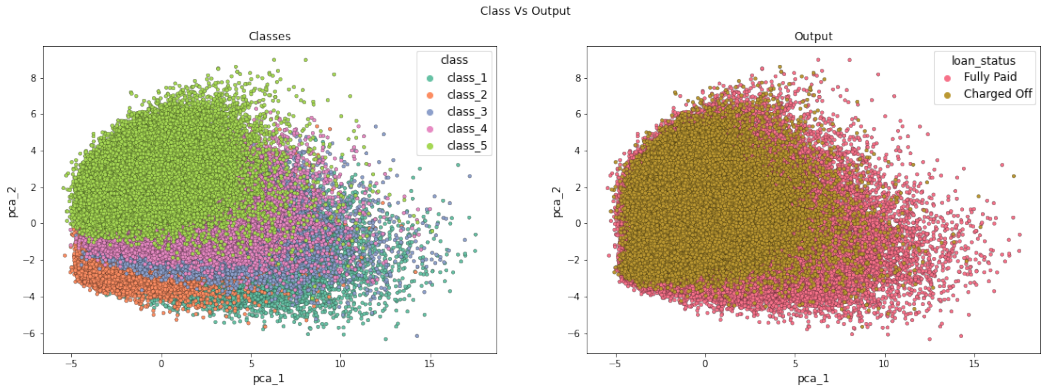


Fonte: Autoria própria

Aplicando o PCA nas entradas para a visualização, encontramos 2 componentes principais que possuem respectivamente $15,87\%$ e $8,27\%$ da variância original, e juntos explicam $24,15\%$ da variância total. É importante ressaltar que a variância total representada é baixa e que, portanto, a visualização do espaço gerado fica prejudicada ao não representar grande parte da variância.

O resultado dos K-means para 5 clusters pode ser visto na figura 47 abaixo, junto com a variável *target*.

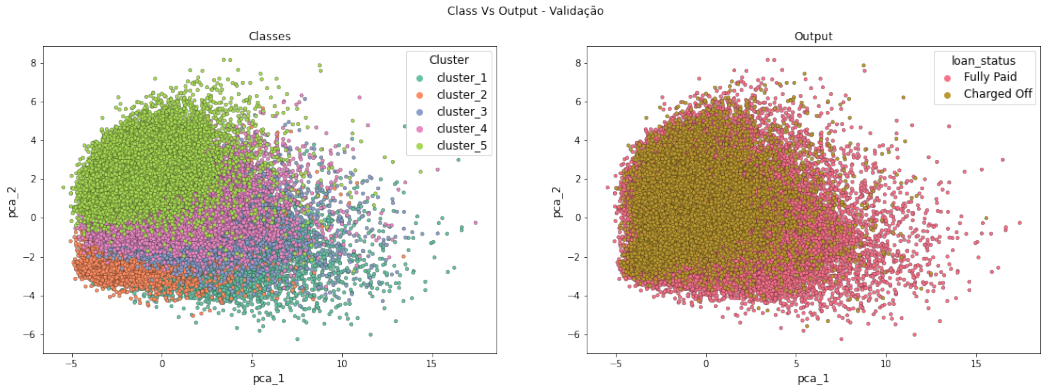
Figura 47: K-means nas entradas para k=5 no cenário 3



Fonte: Autoria própria

Para os dados de validação, o espaço se manteve. A imagem dos clusters com esses dados pode ser vista na figura 48 abaixo.

Figura 48: K-means com k=5 nas entradas cenário 3 validação



Fonte: Autoria própria

4.3.5.1 Análise de resultados

Considerando os grupos encontrados, podemos calcular a inadimplência a partir da quantidade de empréstimos *Charged Off* por grupo. Ao fazer essa conta para os dados de treino, encontramos a inadimplência esperada pelo modelo K-means. Fazendo isso para os dados de validação, encontramos a inadimplência observada em validação. Para calcular o resultado do modelo LC, utilizamos a inadimplência esperada por *grade*, calculando a média dessa inadimplências por grupo e obtendo a inadimplência esperada pelo modelo LC.

Dessa forma, temos a tabela 11:

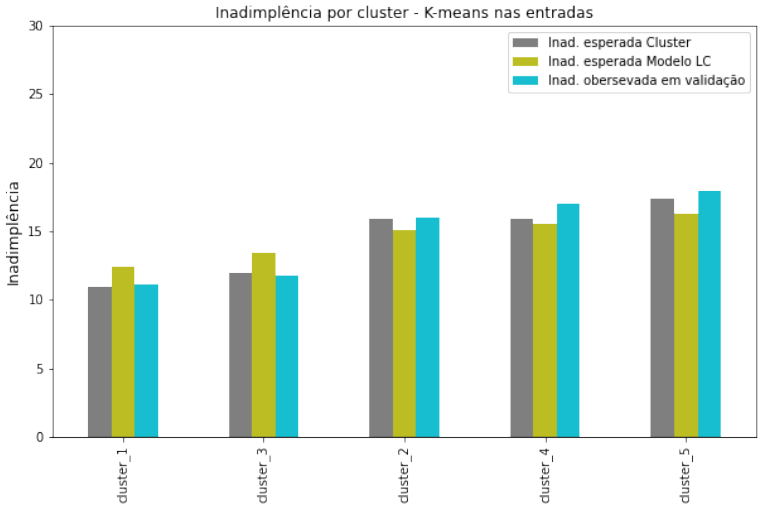
Tabela 11: Inadimplência K-means nas entradas para k=5 no cenário 3

Cluster	Inad. esperada Cluster(%)	Inad. esperada Modelo LC(%)	Inad. observada em validação(%)	Num pontos validação
cluster_1	10,96	12,43	11,12	11598
cluster_2	15,89	15,12	15,98	13285
cluster_3	11,93	13,43	11,76	7021
cluster_4	15,94	15,50	17,00	12640
cluster_5	17,37	16,26	17,96	10384

Fonte: Autoria própria

Outro tipo de visualização é por meio do gráfico 49, em que ordenamos os clusters por inadimplência.

Figura 49: Inadimplência K-means nas entradas para k=5 no cenário 3



Fonte: Autoria própria

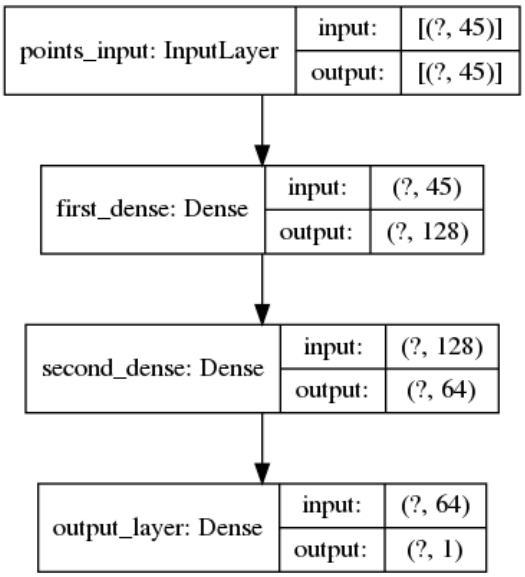
Definindo o erro médio ponderado como a média das diferenças absolutas entre a inadimplência esperada e observada por cluster, ponderada pela quantidade de pontos em valiação, temos que o erro do K-means nas entradas foi de $0,43\%$ e o erro do modelo LC foi de $1,36\%$.

O erro absoluto máximo do modelo LC foi $1,69\%$ no cluster 5, e do modelo K-means nas entradas foi $1,07\%$ no cluster 4.

4.3.6 K-means no espaço de latência

O primeiro passo é treinar uma rede neural. Para isso, utilizamos uma rede de arquitetura de acordo com a figura 50 e com mais detalhes no código 7. Nela, a camada de entrada recebe 45 entradas, a primeira camada escondida é uma *Dense* com 128 neurônios e com função de ativação *relu*, seguida de uma segunda camada escondida *Dense* com 64 neurônios e com a mesma função de ativação *relu*, e por último, uma camada de saída com uma *Dense* de 1 neurônio e com função de ativação sigmoide.

Figura 50: Arquitetura da Rede Neural cenário 3



Fonte: Autoria própria

Código 7: Arquitetura da Rede cenário 3

```
model_input = Input(shape=(45,)
                    ,name='points_input')

first_dense = Dense(128, activation='relu',
            kernel_initializer='he_uniform',
            name='first_dense')(model_input)
second_dense = Dense(64, activation='relu',
            kernel_initializer='he_uniform',
            name='second_dense')(first_dense)
target = Dense(1, activation='sigmoid',
            name='output_layer')(second_dense)

model = Model(inputs=model_input,
              outputs=target)
```

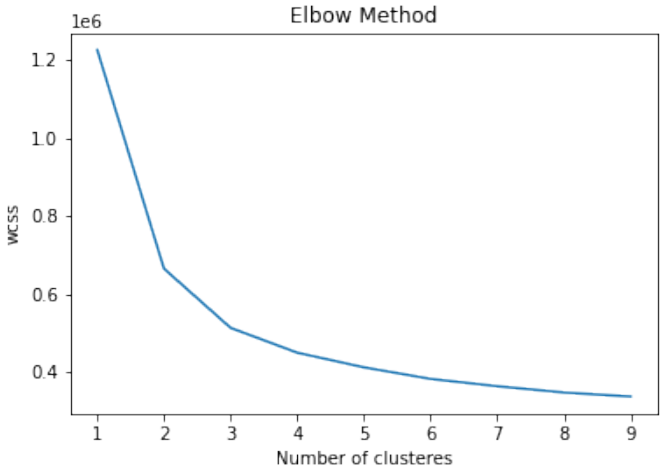
Fonte: Autoria própria

Os parâmetros de função custo e métrica de treinamento foram os mesmos da seção 4.1.5.

A acurácia final do modelo foi de $0,85$, ou seja, o modelo não teve um resultado final satisfatório, uma vez que a proporção de empréstimos *Fully Paid* é de 85%.

Para definir o número de clusters, vamos utilizar o *Elbow Method*. De acordo com o gráfico 51, vemos que 5 é um número bom de clusters.

Figura 51: *Elbow Method* no espaço de latência no cenário 3

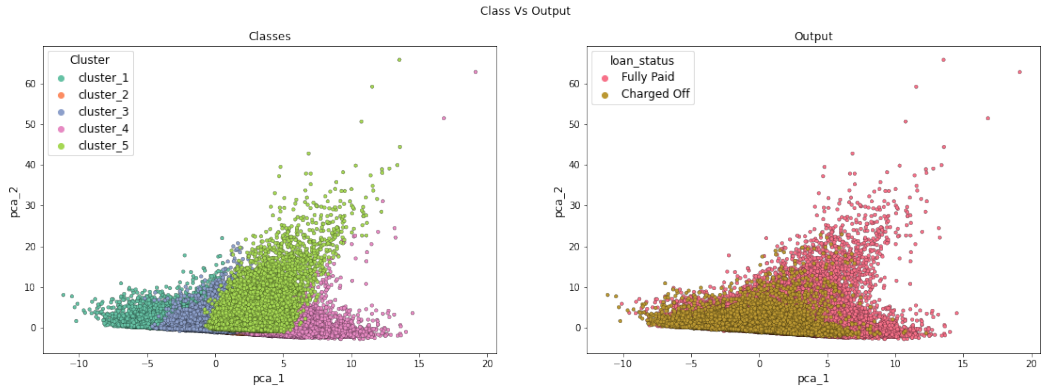


Fonte: Autoria própria

Aplicando o PCA nas entradas para a visualização, encontramos 2 componentes principais que possuem respectivamente $13,37\%$ e $6,85\%$ da variância original, e juntos explicam $20,22\%$ da variância total. É importante ressaltar que a variância total representada é baixa e que, portanto, a visualização do espaço gerado fica prejudicada ao não representar grande parte da variância.

O resultado dos K-means para 5 clusters pode ser visto na figura 52 abaixo, junto com a variável *target*.

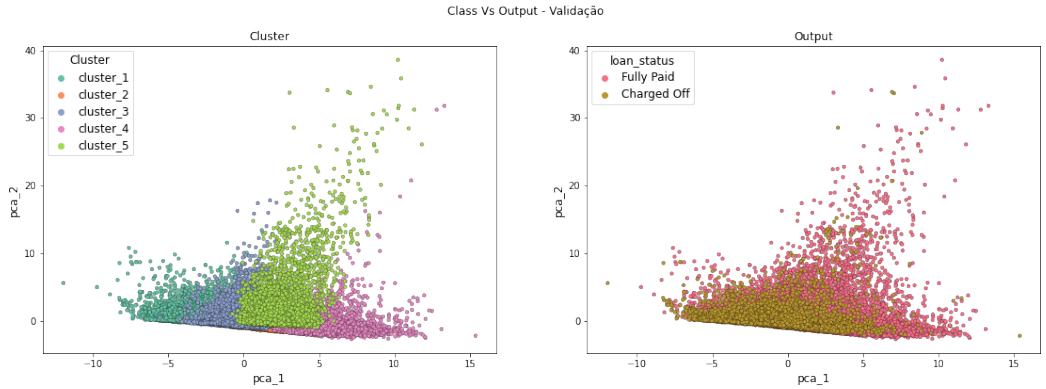
Figura 52: K-means no espaço de latência para k=5 no cenário 3



Fonte: Autoria própria

Para os dados de validação, o espaço se manteve. A imagem dos clusters com esses dados pode ser visto na figura 53 abaixo.

Figura 53: Validação K-means no espaço de latência para k=5 cenário 3



Fonte: Autoria própria

4.3.6.1 Análise de resultados

A partir dos grupos encontrados, vamos novamente calcular a inadimplência esperada pelo modelo K-means no espaço de latência, a observada em validação e a esperada pelo modelo LC. Dessa forma, temos a tabela 12:

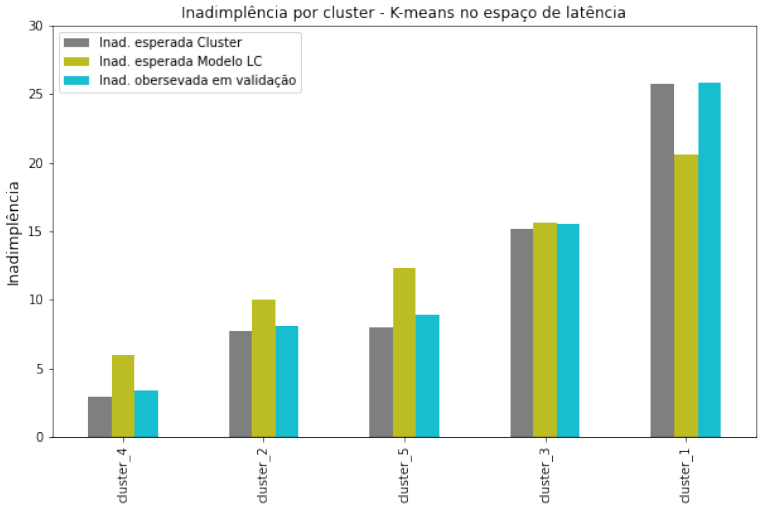
Tabela 12: Inadimplência K-means no espaço de latência para k=5

Cluster	Inad. esperada Cluster(%)	Inad. esperada Modelo LC(%)	Inad. observada em validação(%)	Num pontos validação(%)
cluster_1	25,72	20,58	25,82	14654
cluster_2	7,71	10,00	8,11	10009
cluster_3	15,17	15,62	15,54	18942
cluster_4	2,90	5,95	3,44	5377
cluster_5	7,99	12,37	8,95	5946

Fonte: Autoria própria

Outro tipo de visualização é por meio do gráfico 49, em que ordenamos os clusters por inadimplência.

Figura 54: Inadimplência K-means no espaço de latência para k=5 no cenário 3



Fonte: Autoria própria

Calculado o erro médio ponderado, definido anteriormente, temos que o erro do K-means no espaço de latência foi de $0,38\%$ e o erro do modelo LC foi de $2,39\%$.

O erro absoluto máximo do modelo LC foi $5,23\%$ no cluster 1, e do modelo K-means no espaço de latência foi $0,96\%$ no cluster 5.

Quando a inadimplência esperada é maior que a observada, o rendimento do investidor é maior que o esperado, assim, a taxa de juros cobrada poderia ter sido menor. Na situação inversa, quando a inadimplência esperada é menor que a observada, o rendimento do investidor é menor do que o esperado, então a taxa de juros cobrada deveria ter sido

maior.

Assim, como os empréstimos foram classificados pelo modelo LC, os empréstimos do Cluster 2, 4 e 5 geraram um rendimento maior que o esperado, e empréstimos do cluster 1 produziram um rendimento menor e possivelmente um prejuízo.

Por outro lado, o K-means no espaço de latência tem uma precisão muito melhor, uma vez que a inadimplência esperada é sempre próxima da observada em validação. Isso mostra uma oportunidade de melhoria do modelo LC a partir dos grupos encontrados.

Uma solução é, após a análise de risco do modelo LC, ajustar o *grade* de acordo com o a inadimplência esperada pelo modelo K-means no espaço de latência. Isso resultaria em uma carteira mais equilibrada, em que o investidor precisa de menos emprestimos para ter o rendimento esperado, e as empresas conseguem taxas de juros mais justas em relação ao seu risco.

4.3.7 Comparação de resultados

A abordagem tradicional do K-means nas entradas apresentou um erro médio ponderado semelhante ao K-means no espaço de latência. Porém, os grupos encontrados não se diferem muito entre si em relação à inadimplência, além de serem próximos de 15%, a proporção de inadimplentes da base. Por conta disso, o algoritmo não ajudou na análise do modelo LC, nem apresentou informações novas.

Por outro lado, o K-means no espaço de latência consegue encontrar grupos com uma inadimplência bem diferente entre si, além de apresentar situações em que o modelo LC não performa bem. Ou seja, encontrou grupos voltados para o cenário de crédito e foi possível mapear possibilidades para melhorar o modelo LC combinando com os clusters encontrados.

Mesmo sem todas as informações utilizadas pela *Lending Club*, o K-means no espaço de latência consegue encontrar um bom resultado. Se tivessemos as todas as variáveis, a solução poderia ser ainda melhor.

5 CONSIDERAÇÕES FINAIS

5.1 Conclusão do projeto

O objetivo maior deste trabalho era expor uma técnica de clusterização baseada no espaço de latência de uma rede neural. O grupo acredita que o trabalho cumpre seu papel, uma vez que definimos a técnica, apresentamos seu funcionamento em cenários gerados artificialmente e um cenário real, além de comparar com o tradicional K-means.

5.2 Contribuições

A grande contribuição do trabalho é a própria abordagem de clusterização, que não apresenta limitações em relação à entradas não numéricas e encontra grupos otimizados para o problema. Além disso, ela pode ser aplicada em diversos cenários do mundo real e permitir encontrar soluções que geram valor para o negócio.

5.3 Perspectivas de continuidade

Ao longo do trabalho, utilizamos o k-means como algoritmo de clusterização para comparação. Assim, para entender um pouco mais das vantagens da técnica proposta poderiamos comparar com outros algoritmos de clusterização. Além disso, podemos ainda utilizar outros algoritmos de clusterização no espaço de latência, já que nele pode haver uma distribuição de pontos mais adequada para densidade, por exemplo.

Além disso, identificamos nos cenários artificiais que a clusterização no espaço de latência apresentou uma interpretabilidade do aprendizado da rede neural treinada. Assim, podemos levantar a hipótese de que interpretar os clusters da última camada de latência tem uma forte relação com o aprendizado da rede neural.

REFERÊNCIAS

[1] AHMAD, A.; KHAN, S. S. Survey of state-of-the-art mixed data clustering algorithms. *IEEE Access*, v. 7, p. 31883–31902, 2019.

[2] Wei, M.; Chow, T.; Chan, R. Clustering heterogeneous data with k-means by mutual information-based unsupervised feature transformation. *Entropy*, v. 17, n. 3, p. 1535–1548, 2015.

[3] YANG, B. et al. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In: PRECUP, D.; TEH, Y. W. (Ed.). *Proceedings of the 34th International Conference on Machine Learning*. International Convention Centre, Sydney, Australia: PMLR, 2017. (Proceedings of Machine Learning Research, v. 70), p. 3861–3870. Disponível em: <http://proceedings.mlr.press/v70/yang17b.html>.

[4] Taigman, Y. et al. Deepface: Closing the gap to human-level performance in face verification. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2014. p. 1701–1708.

[5] ALL Lending Club Loan Data. Disponível em: <https://www.kaggle.com/wordsforthewise/lending-club>.

[6] LLOYD, S. P. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, v. 28, p. 129–137, 1982.

[7] MANNING, C. D.; RAGHAVAN, P.; SCHüTZE, H. *Introduction to Information Retrieval*. [S.l.]: Cambridge University Press, 2008. 357-368 p. Web publication at informationretrieval.org.

[8] AUTOENCODERS. Disponível em: <http://deeplearningbook.com.br/introducao-aos-autoencoders/>.

[9] GOYAL, P. *What is the meaning of latent space*. Disponível em: <https://www.quora.com/What-is-the-meaning-of-latent-space>.

[10] Shearer, C. The crisp-dm model: the new blueprint for data mining. *Journal of Data Warehousing*, v. 5, p. 13–2, 2000.

[11] KLUYVER, T. et al. Jupyter notebooks – a publishing format for reproducible computational workflows. In: LOIZIDES, F.; SCHMIDT, B. (Ed.). *Positioning and Power in Academic Publishing: Players, Agents and Agendas: Proceedings of the 20th International Conference on Electronic Publishing*. [S.l.], 2016. p. 87 – 90.

[12] ROSSUM, G. V.; DRAKE, F. L. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. ISBN 1441412697.

[13] PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011.

[14] HARRIS, C. R. et al. Array programming with NumPy. *Nature*, Springer Science and Business Media LLC, v. 585, n. 7825, p. 357–362, set. 2020. Disponível em: <https://doi.org/10.1038/s41586-020-2649-2>.

[15] MCKINNEY, W. Data structures for statistical computing in python. In: WALT, S. van der; MILLMAN, J. (Ed.). *Proceedings of the 9th Python in Science Conference*. [S.l.: s.n.], 2010. p. 51 – 56.

[16] HUNTER, J. D. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, IEEE COMPUTER SOC, v. 9, n. 3, p. 90–95, 2007.

[17] ABADI, M. et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. Software available from tensorflow.org. Disponível em: <https://www.tensorflow.org/>.