**Detecção de anomalias em despesas dos deputados estaduais**

**de São Paulo por meio de K-Means**

Rodolfo Orlando Viana¹\*; Ana Julia Righetto2

1 Avenida Maria Fernandes Cavallari, 3099 – Jardim Cavallari; 17526-341 Marília, São Paulo, Brasil

2 Nome da Empresa ou Instituição (opcional). Titulação ou função ou departamento. Endereço completo (pessoal ou profissional) – Bairro; 00000-000 Cidade, Estado, País

\*Rodolfo Orlando Viana: eu@rodolfoviana.com.br

**Detecção de anomalias em despesas dos deputados estaduais**

**de São Paulo por meio de K-Means**

**Resumo**

O presente trabalho investigou anomalias em gastos de fundos públicos recebidos pelos deputados da Assembleia Legislativa do Estado de São Paulo [Alesp] por meio da "verba de gabinete". Com as alocações de 2022 superando os anos anteriores e alegações de malversação de recuso público feitas por órgãos de controle, torna-se imperativo examinar esses gastos. Empregamos aprendizado de máquina não supervisionado, especificamente a clusterização K-Means com método de inicialização K-Means++, para discernir anomalias nas despesas. Embora não rotulemos conclusivamente as transações como fraudulentas, nossa metodologia oferece um arcabouço para identificar possíveis inconsistências financeiras, auxiliando órgãos de supervisão em suas análises.

**Palavras-chave:** Alesp; recursos públicos; clusterização; K-Means++; aprendizado de máquina não supervisionado.

**Anomaly detection in expenses of state deputies of São Paulo using K-Means**

**Abstract**

This study investigated anomalies in the spending of public funds received by deputies of the São Paulo State Legislative Assembly [Alesp] through the "office allowance". With the 2022 allocations surpassing previous years and allegations of misappropriation of public funds made by oversight bodies, it becomes imperative to examine these expenditures. We employed unsupervised machine learning, specifically the K-Means clustering with the K-Means++ initialization method, to discern anomalies in expenses. While we do not conclusively label transactions as fraudulent, our methodology provides a framework to identify potential financial inconsistencies, assisting oversight bodies in their reviews.

**Keywords:** Alesp; public funds; clustering; K-Means++; unsupervised machine learning

**Introdução**

Cada um dos 94 parlamentares da Assembleia Legislativa do Estado de São Paulo [Alesp] tem direito aos Auxílio-Encargos Gerais de Gabinete de Deputado e Auxílio-Hospedagem, referenciados conjuntamente como “verba de gabinete”. Tal direito foi conferido pela resolução 783, artigo 11, de 1º de julho de 1997. Trata-se de um valor mensal devido pelo Estado aos deputados a fim de que eles possam cobrir gastos com o funcionamento e manutenção dos gabinetes, com hospedagem e demais despesas inerentes ao pleno exercício das atividades parlamentares.

Tais gastos previstos na legislação são agregados em 11 categorias, dentre as quais materiais e serviços gráficos, consultoria, combustíveis, locação de automóveis, hospedagem. Em 2022, considerando resolução 783, de 1º de julho de 19971, que estipula o limite máximo da verba de gabinete em 1.250 unidades fiscais do Estado de São Paulo [Ufesp], e o valor da Ufesp em R$ 31,97, o limite mensal da verba de gabinete que poderia ser ressarcido por deputado no ano passado foi de R$ 39.962,50.

Naquele ano, o valor total empenhado para custeio da verba de gabinete perfez R$ 26.652.243,51. O montante foi 24,43% maior que a soma em 2021, de R$ 21.419.316,88, e menor do que o valor anotado na rubrica para 2023, de R$ 28.607.099,96. Caso este montante se cumpra neste ano, será a primeira vez que o valor ultrapassa R$ 28,5 milhões desde 2018.

Tais somas de recursos públicos podem servir, ainda que parcialmente, para infringir a lei. Um exemplo é o processo investigatório SEI 29.0001.0246360.2021-544, cujo pedido de instauração foi feito pelo Procurador Mario Antonio de Campos Tebet em 5 de maio de 2022. A peça elenca possível malversação no uso da verba de gabinete por parte do deputado estadual Murilo Felix, que a teria empregado para pagar pela locação de imóveis pertencentes a aliados políticos e nunca utilizados.

Com este contexto, o presente trabalho busca ser um instrumento para avaliação de malversação de dinheiro público por meio de aprendizado de máquina não supervisionado. O objetivo desta peça não é afirmar peremptoriamente se determinada despesa é fraudulenta ou não; seu escopo é servir de ferramenta para uma observação inicial dos gastos, que podem ser analisados por meio de clusterização, onde se objetiva encontrar um grupo de despesas cujos valores indicam possíveis anomalias.

**Material e Métodos**

A primeira etapa consistiu na captura dos dados a partir do Portal de Dados Abertos da Alesp, onde estão disponíveis arquivos no formato xml que datam desde 2002 e contêm elementos que indicam o período de referência (“Ano”, “Mês”), além de informações tanto do parlamentar (“Matrícula”, “Deputado”) quanto da despesa (“Fornecedor”, “CNPJ”, “Tipo”, “Valor”). Para este trabalho, foram utilizados apenas “CNPJ” e “Valor”, a fim de desconsiderar eventuais vieses ideológicos. Dado o contexto temporal dos gastos, “Ano” e “Mês” foram usados tão somente para realizar a deflação dos valores até 31 dez. 2022 seguindo o índice de preço ao consumidor amplo [IPCA], conforme divulgado pelo Instituto Brasileiro de Geografia e Estatística [IBGE]. Com isso, descartou-se a temporalidade das despesas.

Foram inseridas neste estudo apenas as despesas relacionadas a alimentação e hospedagem compreendidas entre os anos de 2018 e 2022. Descartaram-se, ainda, fornecedores com menos de 20 despesas no quinquênio, haja vista a necessidade de se ter número significativo para a realização de clusterização.

Uma análise exploratória foi realizada para compreender os dados e sua dispersão no conjunto. No quinquênio observado, foram 4.453 registros de despesas em 86 números de CNPJ únicos, totalizando R$ 1.784.601,08. Cada despesa apresentou valor médio de R$ 400,46, porém com desvio-padrão elevado (R$ 967,47), indicando significativa dispersão dos dados em relação à média. O coeficiente de variação de 241,41% demonstrou alto grau de variabilidade relativo à média.

Notou-se ainda que a média é superior ao terceiro quartil. Isso indica que o conjunto de dados está inclinado para valores mais baixos, apesar da significante presença de outliers que puxa o terceiro quartil para cima. Graficamente, o valor médio maior que o terceiro quartil sugere assimetria positiva: a cauda do lado direito é mais longa do que do lado esquerdo. Essa indicação é corroborada com a assimetria de 5,21, enquanto a curtose de 32,67 comprova cauda longa e picos acentuados em comparação à distribuição normal.

Tabela 1. Estatísticas dos dados analisados

|  |  |
| --- | --- |
| Medida | Valor |
| Contagem | 4.453 |
| Média (R$) | 400,763773 |
| Desvio-padrão (R$) | 967,469752 |
| Mínimo (R$) | 6,49 |
| 1º Quartil (R$) | 55,75 |
| 2º Quartil (R$) | 123,14 |
| 3º Quartil (R$) | 276,18 |
| Máximo (R$) | 10.250,41 |
| Coeficiente de variação (%) | 241,40648... |
| ­­Assimetria | 5,21061... |
| Curtose | 32,66851... |

Fonte: Dados originais da pesquisa

Em seguida, foi construído um algoritmo de clusterização por K-Means. Nesta técnica, a organização dos conjuntos é feita com a determinação aleatória de um centroide, um ponto que observa a distância euclidiana dos demais dados em relação a ele, de acordo com MacQueen (1967). Dado um conjunto de observações , o algoritmo de K-Means reparte as n observações em conjuntos a fim de minimizar a soma dos quadrados dentro do cluster. Seu objetivo é encontrar

(1)

onde, : média dos pontos em .

A quantidade de agrupamentos a serem utilizados pelo algoritmo deve ser conhecida a priori. O método do cotovelo — Elbow method —, apresentado por Joshi e Nalwade (2012), é uma forma de obter esse número com base na iteração entre possíveis centros de clusters e a soma dos quadrados das distâncias entre eles e os pontos de dados. A heurística opera sob a lógica de que, ao aumentar o número de agrupamentos, ocorrerá a diminuição da soma dos quadrados intracluster, haja vista a maior proximidade dos pontos em relação aos centroides de seus respectivos agrupamentos. Em determinado momento, o valor de tal diminuição se tornará marginal — traduzido de maneira visual em gráfico, uma linha teria inicialmente quedas acentuadas para, em seguida, se estabilizar na posição horizontal, formando um "cotovelo" (Figura 1). O ponto em que essa estabilização se torna perceptível representa uma estimativa do número ideal de clusters.

Figura 1. Método do cotovelo para o CNPJ 02.530.487/0002-20, selecionado aleatoriamente

Fonte: Dados originais da pesquisa

Nota: \*neste exemplo, o método indica o uso de quatro clusters

A determinação do número de clusters em si, porém, não garante que o algoritmo encontre os melhores pontos para servirem de centroides. A alta sensibilidade da técnica de agrupamento K-Means pode levar a uma solução de mínimo local em vez de uma global, levando a partições que não sejam ideais, segundo Morissette e Chartier (2013). Para sobrepor essa limitação, este trabalho se utilizou do método de inicialização K-Means++, em que o centroide é atualizado com base na distância entre os pontos de dados e o centroide anteriormente selecionado, conforme sugerido por Arthur e Vassilvitskii (2007). A mudança sucessiva entre centroides reduz as chances de o algoritmo K-Means convergir para uma solução abaixo do ideal.

Dado um conjunto de pontos D e um conjunto de centroides já selecionados C, a probabilidade de se escolher o ponto de dado x como próximo centroide é calculada por meio de

(2)

sendo, : distância entre o ponto e o centroide mais próximo em .

Com os centroides inicializados, cada ponto é atribuído ao centroide mais próximo. Tais pontos de dados próximos ao centroide formam clusters. Considerando o ponto x e um conjunto de centroides C, o rótulo do cluster l ao qual x pertence é computado por

(3)

Em seguida, cada centroide é recalculado tomando a média da distância de todos os pontos a eles atribuídos,

(4)

onde, : conjunto de todos os pontos de dados atribuídos ao centroide i.

A cada iteração de atualização de centroides é computado o custo da função — a inércia. Para conjunto univariado, a operação segue

(5)

onde, : centroide do cluster para o qual o ponto foi atribuído.

O algoritmo desenvolvido também adotou critérios de convergência avançados ao comparar o movimento dos centroides entre iterações. Sendo o conjunto de centroides na iteração , o algoritmo converge se

(6)

onde, tol: tolerância especificada; : distância euclidiana.

A validação dos resultados obtidos a partir da implementação dessas técnicas foi realizada com duas medidas. O primeiro é o método da silhueta — Silhouette method —, seguindo o trabalho de Rousseeuw (1987). Esta técnica observa a similaridade de um ponto com seu cluster em comparação com outros clusters a partir de

(7)

onde, : a distância média de i para todos os outros pontos intracluster; : a menor distância média de i para todos os pontos em clusters diferentes.

O método da silhueta retorna resultados no intervalo de -1 a 1. Se o valor for próximo de -1, significa que o ponto está agrupado de maneira errada; próximo de 0, o ponto está entre dois clusters, de forma que o agrupamento pode ser aprimorado; próximo de 1, o ponto está bem agrupado.

Enquanto o método da silhueta faz comparação entre um ponto único e os agrupamentos, o índice de Davies-Bouldin (1979), segunda medida usada na validação dos resultados, observa a coesão de do cluster, dada a lógica de que um agrupamento adequado é denso em si, ao passo que distante dos demais agrupamentos. Melhor o agrupamento quanto mais o índice se aproxima de 0, resultado obtido por

(8)

sendo, k: número de clusters; i,j: clusters diferentes; : dispersão interna dos clusters i e j, respectivamente; : a distância entre clusters i e j.

**Resultados e Discussão**

O título da seção Resultados e Discussão deve ser alinhado à esquerda, grafado em negrito com as primeiras letras das palavras em letras maiúsculas. É permitido que a seção seja dividida em subtópicos com formatação de acordo com a descrição no item 1.1 Formato e margens, apresentados na mesma ordem da seção Material e Métodos. Nesta seção devem ser apresentados, discutidos e interpretados os resultados obtidos no trabalho, ou seja, autores devem fazer uma discussão comparativa dos resultados do seu trabalho com aqueles existentes na literatura e elaborar uma análise crítica dos dados, destacando as limitações e pontos positivos dos resultados.

**Considerações Finais**

O título da seção Conclusão(ões) ou Considerações Finais deve ser alinhado à esquerda e grafado em negrito. Fica a critério do aluno e do orientador a escolha de qual termo melhor se adequa ao trabalho. Esta seção deve conter frases curtas, apresentando as conclusões e inferências elaboradas a partir da discussão dos resultados. É importante que estas frases não sejam meras reproduções dos resultados, respondendo aos objetivos propostos no trabalho. Os autores não devem, em hipótese alguma, mencionar, citar ou reproduzir resultados de outros estudos na(s) conclusão(ões) ou considerações finais do TCC. Por fim, salienta-se que essa seção não deve conter tabelas ou figuras, sendo redigida de forma sucinta.

**Agradecimento**

Agradeço a Pedro Orlando, meu afilhado de seis anos que, com seus convites para assistir a vídeos do Enaldinho ou fazer um piquenique na sala de casa, conseguiu me distrair deste trabalho ao tempo que recarregou minhas energias para nele prosseguir.

**Referências**

Arthur, D.; Vassilvitskii, S. 2007. K-Means++: The advantages of careful seeding. Proceedings of Annual ACM-SIAM Symposium on Discrete Algorithms: 1027-1035.

Assembleia Legislativa do Estado de São Paulo [Alesp]. 1997. Resolução n. 783, de 1° de julho de 1997. Altera a Resolução n° 776, de 14/10/1996, que implantou a nova estrutura administrativa, cria o Núcleo de Qualidade e institui a verba de gabinete. Disponível em: https://www.al.sp.gov.br/repositorio/legislacao/resolucao.alesp/1997/original-resolucao.alesp-783-01.07.1997.html. Acesso em: 19 março 2023.

Davies, D.L.; Bouldin, D.W. 1979. A cluster separation measure. IEEE Transactions on Pattern Analysis and Machine Intelligence 2: 224–227.

Joshi, K.D.; Nalwade, P.S. 2012. Modified K-Means for better initial cluster centres. International Journal of Computer Science and Mobile Computing 7: 219-223.

MacQueen, J. 1967. Some methods for classification and analysis of multivariate observations. In: 5th Berkeley Symposium on Mathematical Statistics and Probability, 1967, Los Angeles, LA, Estados Unidos, Anais… p. 281-297.

Ministério Público de São Paulo. 2022. Sistema Eletrônico de Informações. Disponível em: https://www.mpsp.mp.br/sei-sistema-eletronico-de-informacoes Acesso em: 26 março 2023.

Morissette, L.; Chartier, S. 2013. The K-Means clustering technique: General considerations and implementation in Mathematica. Tutorials in Quantitative Methods for Psychology 9: 15-24.

Rousseeuw, P.J. 1987. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics 20: 53-65.

Secretaria da Fazenda e Planejamento do Governo do Estado de São Paulo. 2023. Execução orçamentária e financeira. Disponível em: https://www.fazenda.sp.gov.br/sigeolei131/paginas/flexconsdespesa.aspx. Acesso em: 19 março 2023.

Secretaria da Fazenda e Planejamento do Governo do Estado de São Paulo. 2023. Índices. Disponível em: https://portal.fazenda.sp.gov.br/Paginas/Indices.aspx. Acesso em: 26 março 2023.

**Apêndice**

1. **Código-fonte do algoritmo**

from typing import Tuple

import numpy as np

class KMeans:

"""

k-means com critérios de convergência aprimorados.

Atributos:

k (int): Número de clusters.

max\_iters (int): Número máximo de iterações para o k-means.

tol (float): Tolerância de convergência baseada no movimento do

centroide.

n\_init (int): Número de vezes que o algoritmo será executado com

diferentes seeds de centroides.

threshold (int): Percentil para detecção de anomalias.

centroids (np.ndarray): Centroides para os clusters.

"""

def \_\_init\_\_(

self,

k: int = 2,

max\_iters: int = 100,

tol: float = 1e-4,

n\_init: int = 30,

threshold: int = 95,

centroids: np.ndarray = None,

):

"""

Inicialização com parâmetros especificados.

"""

self.k = k

self.max\_iters = max\_iters

self.tol = tol

self.n\_init = n\_init

self.threshold = threshold

self.centroids = centroids

@staticmethod

def \_kpp\_init(data: np.ndarray, k: int) -> np.ndarray:

"""

Inicializa os centroides usando o método k-means++.

Argumentos:

data (np.ndarray): Dados de entrada.

k (int): Número de centroides desejados.

Retorna:

centroids (np.ndarray): Centroides inicializados.

"""

centroids = [data[np.random.choice(len(data))]]

for \_ in range(1, k):

squared\_dist = np.array(

[np.min([

np.linalg.norm(c - x) \*\* 2 for c in centroids

]) for x in data]

)

probs = squared\_dist / squared\_dist.sum()

centroid = data[np.argmax(probs)]

centroids.append(centroid)

return np.array(centroids)

def get\_optimal\_k(self, data: np.ndarray, k\_max: int = 10) -> int:

"""

Aplica método Elbow para obter o número de clusters ideal.

Argumentos:

data (np.ndarray): Dados usados no algoritmo K-Means.

k\_max (int): Número máximo de clusters. Valor-padrão: 10.

Retorna:

optimal\_k (int): Número de clusters ideal.

"""

sum\_sq = []

for k in range(1, k\_max + 1):

self.k = k

self.fit(data)

inertia = np.sum(

[

np.linalg.norm(

data[i] - self.centroids[self.labels[i]]

) \*\* 2

for i in range(len(data))

]

)

sum\_sq.append(inertia)

diffs = np.diff(sum\_sq, 2)

optimal\_k = np.argmin(diffs) + 1

return optimal\_k

def \_single\_run(self, data: np.ndarray) -> Tuple[np.ndarray, np.ndarray, float]:

"""

Realiza execução única do algoritmo k-means.

Argumentos:

data (np.ndarray): Dados de entrada.

Retorna:

centroids (np.ndarray): Melhores centroides após a execução

do k-means.

labels (np.ndarray): Atribuições de cluster para cada ponto

de dado.

inertia (float): Distância total dos pontos de dados a

partir de seus centroides atribuídos.

"""

centroids = self.\_kpp\_init(data, self.k)

for \_ in range(self.max\_iters):

dist = np.linalg.norm(data[:, np.newaxis] - centroids, axis=2)

labels = np.argmin(dist, axis=1)

new\_centroids = np.array(

[data[labels == i].mean(axis=0) for i in range(self.k)]

)

if np.all(np.abs(new\_centroids - centroids) < self.tol):

break

centroids = new\_centroids

inertia = np.sum(

[

np.linalg.norm(data[i] - centroids[labels[i]]) \*\* 2

for i in range(len(data))

]

)

return centroids, labels, inertia

def fit(self, data: np.ndarray) -> None:

"""

Ajusta o algoritmo k-means aos dados.

Argumento:

data (np.ndarray): Dados de entrada.

"""

min\_inertia = float("inf")

best\_centroids = None

best\_labels = None

for \_ in range(self.n\_init):

centroids, labels, inertia = self.\_single\_run(data)

if inertia < min\_inertia:

min\_inertia = inertia

best\_centroids = centroids

best\_labels = labels

self.centroids = best\_centroids

self.labels = best\_labels

def detect(self, data: np.ndarray) -> np.ndarray:

"""

Detecta anomalias nos dados com base na distância ao centroide

mais próximo.

Argumentos:

data (np.ndarray): Dados de entrada.

Retorna:

anomalies (np.ndarray): Anomalias detectadas.

"""

dist = np.min(

np.linalg.norm(

data[:, np.newaxis] - self.centroids, axis=2

), axis=1

)

threshold = np.percentile(dist, self.threshold)

anomalies = data[dist > threshold]

return anomalies

def get\_labels(self, data: np.ndarray) -> np.ndarray:

"""

Atribui cada ponto de dado ao centroide mais próximo para

determinar seu cluster.

Argumento:

data (np.ndarray): Conjunto de dados.

Retorna:

labels (np.ndarray): Array de labels de cluster

correspondentes a cada ponto de dado.

"""

dist = np.linalg.norm(data[:, np.newaxis] - self.centroids, axis=2)

labels = np.argmin(dist, axis=1)

return labels

class Score:

"""

Cálculo de scoring para algoritmo de clusterização.

"""

@staticmethod

def silhouette(data: np.ndarray, labels: np.ndarray) -> float:

"""

Calcula o score do método da silhueta.

Argumentos:

data (np.ndarray): Dados de entrada.

labels (np.ndarray): Atribuições de cluster para cada ponto

de dado.

Retorna:

float: Silhouette Score calculado.

"""

unique\_labels = np.unique(labels)

silhouette\_vals = []

for index, label in enumerate(labels):

same\_cluster = data[labels == label]

a = np.mean(np.linalg.norm(same\_cluster - data[index], axis=1))

other\_clusters = [

data[labels == other\_label]

for other\_label in unique\_labels

if other\_label != label

]

b\_vals = [

np.mean(np.linalg.norm(cluster - data[index], axis=1))

for cluster in other\_clusters

]

b = min(b\_vals)

silhouette\_vals.append((b - a) / max(a, b))

return np.mean(silhouette\_vals)

@staticmethod

def daviesbouldin(data: np.ndarray, labels: np.ndarray) -> float:

"""

Calcula o índice de Davies-Bouldin.

Argumentos:

data (np.ndarray): Dados de entrada.

labels (np.ndarray): Atribuições de cluster para cada ponto

de dado.

Retorna:

float: Davies-Bouldin Score calculado.

"""

unique\_labels = np.unique(labels)

centroids = np.array(

[data[labels == label].mean(axis=0) for label in unique\_labels]

)

avg\_dist\_within\_cluster = np.array(

[

np.mean(

np.linalg.norm(

data[labels == label] - centroids[label], axis=1

)

)

for label in unique\_labels

]

)

centroid\_dist = np.linalg.norm(

centroids[:, np.newaxis] - centroids, axis=2

)

np.fill\_diagonal(centroid\_dist, float("inf"))

cluster\_ratios = (

avg\_dist\_within\_cluster[:, np.newaxis] + avg\_dist\_within\_cluster

) / centroid\_dist

max\_cluster\_ratios = np.max(cluster\_ratios, axis=1)

return np.mean(max\_cluster\_ratios)