UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ARTES, CIÊNCIAS E HUMANIDADES
SISTEMAS DE INFORMAÇÃO

Relatório 2: Pokemon With Stats

Prof. Dra. Ana Amélia Benedito Silva

GRUPO 8 TURMA 02

Lauro Hiroshi Pimentel Masuda – 9875437

Silas Fernandes Moreira – 9761718

Victor Gomes de O. M. Nicola – 9844881

São Paulo

2017

LAURO HIROSHI PIMENTEL MASUDA
SILAS FERNANDES MOREIRA
VICTOR GOMES DE O. M. NICOLA

Relatório 2: Pokemon with stats

Trabalho apresentado em cumprimento às exigências da disciplina de Métodos Quantitativos Aplicados à Administração de Empresas I, ministrada pela Profª Dra. Ana Amélia.

São Paulo
2017

RESUMO

Com o conjunto de dados *Pokemon with stats*, que contém atributos de todos os Pokémon da primeira à sexta geração dos jogos de Pokémon para *gameboy*, o presente trabalho buscou encontrar relações entre atributos de combate e outros atributos específicos por meio de métodos de clusterização. Adicionou-se um novo atributo, retirado do conjunto: *List of Pokémon by Evolution Family*; o atributo evolução. Os Pokémon foram separados em três categorias (evolução 1, 2 e 3[1]) ou quatro (evolução 1, 2, 3 e lendários/Mega).

Aplicando PAM para K = 3 e K = 4 e comparando os resultados obtidos com os esperados, obteve-se uma taxa de acerto de 71,88% e 66,50%, respectivamente.

Foi aplicada uma técnica de medição de importância das variáveis, chamada Boruta, a fim de determinar quais variáveis influenciam mais na determinação da variável evolução. O resultado permitiu realizar uma análise mais profunda e os resultados finais indicaram que ao retirar os três atributos menos relevantes, obteve-se a maior taxa de acerto quando K = 3, que foi de 77.00%.

[1] Os Pokémon lendários e Mega foram agrupados junto com os de evolução 3.

INTRODUÇÃO

O trabalho aqui apresentado utiliza um conjunto de dados composto por 800 observações de 721 espécies Pokémon que representam todas as espécies da primeira à sexta geração da série de jogos para *gameboy* criados pela *Game Freak* e atualmente pertencentes à *The Pokemon Company*.

Este conjunto de dados contém todos os atributos[2] de combate dos Pokémon nos jogos. São variáveis quantitativas discretas: Pontos de Vida (*HP*), Ataque (*Attack*), Defesa (*Defense*), Ataque Especial (*Sp. Atk*), Defesa Especial (*Sp. Def*), Velocidade (*Speed*) e *Total* (soma de todos os atributos anteriores). Além dos atributos de combate, também existem as variáveis quantitativas geração (*Generation*) e *ID* (número de identificação do Pokémon), a variável booleana Lendário (*Legendary*) e três variáveis qualitativas nominais: Nome (*Name*), Tipo 1 (*Type 1*) e Tipo 2 (*Type 2*)[3].

Em posse destes atributos, o objetivo deste trabalho foi determinar se, com alguns atributos seria possível fazer asserções acerca de outras variáveis específicas, como Geração, Tipo 1 e/ou 2, etc. e quais destes atributos seriam mais relevantes para tais afirmações. Para isto, foram utilizadas técnicas de clusterização e um método de seleção de variáveis.

[2] Detalhados no relatório 1.
[3] Posteriormente, a variável Evolução (*Evolution*) foi inserida conforme detalhado em outras seções do trabalho

SUMÁRIO

1. METODOLOGIA

A análise do conjunto de dados foi realizada, principalmente, pelo método de particionamento em clusters PAM (*Partitioning Around Medoids)* e um método de seleção de variáveis descritos nos subtópicos seguintes. Porém, outras duas técnicas de clusterização foram estudadas e testadas com o intuito de escolher a melhor aplicação possível para esta base. Toda a implementação das funções foi feita utilizando a ferramenta RStudio, versão 1.0.153.

1.1 *k-Means Clustering* e Fanny

k-Means Cluster e Fanny são duas técnicas de clusterização que utilizam conceitos diferentes, mas ambas agrupam os dados mais similares de acordo com seus critérios. A função Clara é uma função otimizada para grandes conjuntos de dados que aplica o método PAM, descrito na subseção (1.2).

Os métodos de clusterização são técnicas de aprendizagem de máquina não supervisionadas. Diferente de classificação, clusterização e outros métodos não supervisionados, não dependem de classes pré-definidas e exemplos de treino com classes rotuladas. Por este motivo, clusterização é uma forma de aprendizagem por observação e não de aprendizagem por exemplos.

O problema de clusterização é, computacionalmente, NP-difícil e o melhor resultado possível seria obtido testando exaustivamente todas as possibilidades de clusters, mas os algoritmos seriam demasiadamente demorados, portanto, os algoritmos mais comuns de clusterização seguem alguma heurística já bem disseminada para tentar buscar soluções ótimas ou o mais próximo possível disso. Esta foi a principal motivação da escolha de analisar com cuidado as ferramentas disponíveis no R antes de iniciar os testes com o conjunto de dados.

Definições

k-Means Clustering é um método de quantização vetorial[4] que particiona *n* observações em *k* clusters, nos quais cada observação pertence ao cluster com a média mais próxima, tornando-se um protótipo do cluster. A medida de similaridade neste algoritmo é uma medida de distância em relação ao ponto médio do cluster, ou seja, os pontos mais próximos do ponto médio do cluster pertencem ao mesmo agrupamento.

O método mais comum do *k-Means Clustering* é o método de Forgy, que pode ser simplificado em 4 passos:

1. Inicialmente, *k* "pontos médios" são escolhidas de forma aleatória;
2. *k* clusters são criados associando cada observação com a média mais próxima.
3. O centróide de cada cluster torna-se a nova média
4. Os passos dois e três são repetidos até que a convergência seja atingida.

Fanny é uma função do R que aplica o método de *fuzzy clustering* ou *soft clustering* a um conjunto de dados. Trata-se de um método de clusterização em que cada ponto do conjunto pode pertencer a mais de um cluster. Os clusters são identificados por meio de medidas de similaridade. Estas medidas podem ser distância, conectividade ou intensidade. A medida escolhida para este trabalho foi a distância euclidiana, pois a maioria dos atributos utilizados são variáveis quantitativas discretas.

Neste método, notas de associação (tags) são designadas a cada ponto do conjunto de dados. Estas tags indicam o grau de quanto cada ponto pertence a cada cluster. Pontos nos extremos possuem menores notas de

[4] Quantização vetorial é uma técnica clássica de quantização de processamento de sinais que permite modelar a função de densidade de probabilidade por meio da distribuição de vetores prototipados

associação e podem estar no mesmo cluster, mas em menor grau, do que os pontos mais próximos do centro.

Implementações

Para utilizar o método K-means no RStudio é necessário importar a biblioteca “stats”, que contém diversas funções para análise estatística. Na figura 1 observa-se os parâmetros da função kmeans.

```
kmeans(x, centers, iter.max = 10, nstart = 1,
       algorithm = c("Hartigan-Wong", "Lloyd", "Forgy",
                     "MacQueen"), trace=FALSE)
```

Figura 1: assinatura da função kmeans do RStudio. Retirado de: https://www.rdocumentation.org/packages/stats/versions/3.4.3/topics/kmeans

Os elementos da função são:

- **x**: conjunto de dados analisado;
- **centers**: é possível definir um conjunto de pontos como centros ou o número de centros buscados. Caso o parâmetro seja um número, é definido um conjunto de centróides aleatoriamente;
- **iter.max**: número máximo de iterações permitidas. Definido por padrão como dez.
- **nstart**: se o centro for um número, este parâmetro serve para definir quantos conjuntos serão escolhidos aleatoriamente
- **algorithm**: métodos heurísticos que podem ser utilizados.
- **trace**: é necessário apenas no método “Hartigan-Wong”. Quando verdadeiro, este parâmetro produz informações sobre o rastro de mudanças de centróides.

Na execução dos testes preliminares utilizou-se esta função com dois parâmetros: x e centers. Como a ideia inicial seria dividir em tipos, o valor escolhido para o parâmetro centers foi 18. A heurística padrão utilizada na função é o método de “Hartigan-Wong” que foi o utilizado neste trabalho.

O método Fanny foi implementado à partir da biblioteca “cluster”, que contém diversos métodos relacionados à clusterização de dados e funções de machine learning. A figura 2 mostra os parâmetros da função fanny.

```
fanny(x, k, diss = inherits(x, "dist"), memb.exp = 2,
      metric = c("euclidean", "manhattan", "SqEuclidean"),
      stand = FALSE, iniMem.p = NULL, cluster.only = FALSE,
      keep.diss = !diss && !cluster.only && n < 100,
      keep.data = !diss && !cluster.only,
      maxit = 500, tol = 1e-15, trace.lev = 0)
```

Figura 2: assinatura da função fanny do RStudio. Retirada de: https://www.rdocumentation.org/packages/cluster/versions/2.0.6/topics/fanny

Os elementos da função fanny são:

- **x**: matriz ou janela de dados, ou matriz de dissimilaridade, dependendo do argumento diss.
- **k**: Inteiro que representa o número de clusters, com 0 < k < n/2 onde n é o número de observações.
- **diss**: é um argumento lógico que, quando verdadeiro, x é assumido como matriz de dissimilaridade. Quando falso, x é tratado como uma matriz de observações de variáveis.
- **memb.exp**: número estritamente maior que 1, especificando o expoente de associação usado no critério ajustado. O valor utilizado por padrão é 2.
- **metric**: string que especifica qual métrica a ser utilizada para calcular as dissimilaridades entre as observações. As opções são: distância euclidiana, de Manhattan e de quadrados euclidianos.
- **stand**: quando verdadeiro, os valores de x são normalizados antes de calcular as dissimilaridades. É falso, por padrão.

- **iniMem.p**: matriz n X k ou NULL (por padrão) que pode ser utilizada para especificar uma matriz de associação inicial.
- **cluster.only**: lógico. Quando verdadeiro, nenhuma informação de silhueta será computada.
- **keep.diss, keep.data**: valores lógicos indicando se as dissimilaridades e dados de entrada x devem ser mantidos no resultado. Quando este argumento é colocado como falso, há economia no tempo de alocação de memória.
- **maxit, tol**: número máximo de iterações e tolerância máxima para convergência do algoritmo fanny.
- **trace.lev**: inteiro especificando um nível de rastreamento para impressão de diagnóstico durante o algoritmo C-interno.

As duas técnicas apresentadas acima, não fizeram parte da análise final e, portanto, não entram na seção de apresentação dos resultados. No entanto, o estudo e entendimento destas técnicas foi crucial para o desenvolvimento do trabalho e por isso foi considerado interessante adicionar os conceitos aprendidos.

1.2 PAM (*Partitioning Around Medoids*)

Definição

PAM ou Particionamento ao redor de medóides (traduzido do inglês) é também uma técnica de machine learning não supervisionada. Segundo HAN e KAMBER (Data mining, 2006, p. 406), traduzido:

> "*Ele tenta determinar k partições para n objetos. Depois de uma seleção inicial aleatória de k objetos representativos, o algoritmo repetidamente tenta fazer uma escolha melhor de objetos representativos. Todos os possíveis pares de objetos são analisados, onde um objeto em cada par é considerado um objeto representativo e o outro, não.*"

Assim, um algoritmo simplificado para o processo de clusterização utilizado no PAM é:

1. Arbitrariamente, escolher k objetos no conjunto de observações para serem os objetos representativos;
2. Atribua a cada objeto restante ao cluster com o objeto representativo mais próximo;
3. Selecione aleatoriamente um objeto não-representativo;
4. Calcule o custo total S de trocar o objeto representativo com o não-representativo selecionado;
5. Se S < 0 então troque o objeto representativo atual pelo não representativo selecionado para formar um novo conjunto de k objetos representativos;
6. repita a partir do segundo passo até que não haja mudança no custo S.

A função clara nada mais é do que a função pam otimizada para conjuntos de dados com milhares de observações e não foi escolhida devido ao tamanho do conjunto de dados ser inferior a mil.

Implementação

A implementação do PAM, assim como fanny necessita da biblioteca “cluster” para ser utilizada no RStudio. A figura 3 mostra os parâmetros da função pam.

```
pam(x, k, diss = inherits(x, "dist"), metric = "euclidean",
    medoids = NULL, stand = FALSE, cluster.only = FALSE,
    do.swap = TRUE,
    keep.diss = !diss && !cluster.only && n < 100,
    keep.data = !diss && !cluster.only,
    pamonce = FALSE, trace.lev = 0)
```

Figura 3: assinatura da função pam no RStudio. Retirada de: https://www.rdocumentation.org/packages/cluster/versions/2.0.6/topics/pam

Os argumentos da função são:

- **x**: matriz ou janela de dados, ou matriz de dissimilaridade, dependendo do argumento diss.

- **k**: Inteiro que representa o número de clusters, com 0 < k < n onde n é o número de observações.
- **diss**: é um argumento lógico que, quando verdadeiro, x é assumido como matriz de dissimilaridade. Quando falso, x é tratado como uma matriz de observações de variáveis.
- **metric**: string que especifica a métrica utilizada para calcular as distâncias de dissimilaridades. Pode ser utilizada a distância euclidiana[5] ou de manhattan[6].
- **medoids**: NULL ou um vetor de tamanho k de índices inteiros, de 1 à n, especificando os medoides iniciais.
- **stand**: quando verdadeiro, os valores de x são normalizados antes de calcular as dissimilaridades. É falso, por padrão.
- **cluster.only**: lógico. Quando verdadeiro, nenhuma informação de silhueta será computada.
- **do.swap**:
- **keep.dis, keep.data**: valores lógicos indicando se as dissimilaridades e dados de entrada x devem ser mantidos no resultado. Quando este argumento é colocado como falso, há economia no tempo de alocação de memória.
- **pamonce**: lógico ou inteiro de 0 a 2 que especifica atalhos no algoritmo.
- **trace.lev**: inteiro especificando um nível de rastreamento para impressão de diagnóstico durante o algoritmo C-interno.

Por que PAM e não *K means*, Fanny ou Clara?

A escolha do PAM ocorreu, principalmente, por dois motivos: minimiza a soma das dissimilaridades sendo assim mais robusta e por ser menos sensível a outliers. O segundo motivo é mais vantajoso no conjunto de dados analisado, pois os Pokémon lendários e mega são, em muitos casos, considerados outliers.

[5] Soma dos quadrados das diferenças.
[6] Soma das diferenças absolutas

1.3 Método do cotovelo

Definição

O método do cotovelo é uma interpretação e validação da consistência em análises de clusters. Aplicar este método significa olhar para a variância em função do número de clusters.

Adiciona-se um número arbitrário de clusters no gráfico de maneira que adicionar mais clusters não melhor o modelo. Os primeiros clusters acrescentarão bastante informação, mas em algum ponto, o ganho marginal cairá, formando um ângulo no gráfico. O número de clusters é escolhido neste ponto, por isso o nome de método do cotovelo.

Implementação

A função utilizada para determinar o valor k ideal foi fviz_nbclust, da biblioteca “factoextra”. A figura 4 mostra a assinatura da função no RStudio.

```
fviz_nbclust(x, FUNcluster = NULL, method = c("silhouette", "wss",
  "gap_stat"), diss = NULL, k.max = 10, nboot = 100,
  verbose = interactive(), barfill = "steelblue", barcolor = "steelblue",
  linecolor = "steelblue", print.summary = TRUE, ...)
```

Figura 4: função *fviz_nbclust* com todos os parâmetros possíveis. Retirada de: https://www.rdocumentation.org/packages/factoextra/versions/1.0.5/topics/fviz_nbclust

Os argumentos da função são:

- **x**: matriz numérica ou janela de dados;
- **FUNcluster**: método de particionamento utilizado. Métodos aceitos: kmeans, pam, clara, fanny, etc.

- **method**: método a ser utilizado para determinar o número ideal de clusters. Os métodos possíveis são: silhueta[7], wss e gap_stat;
- **diss**: aplica a função dist() para calcular as distâncias necessárias em cada método do argumento anterior;
- **k.max**: número máximo de cluster a ser considerado;
- **nboot**: usada somente para determinar o número de cluster com gap_stat (não foi utilizada neste trabalho);
- **verbose**: se verdadeira, o resultado do progresso é apresentado na tela;
- **barfill, barcolor**: cores para as barras;
- **linecolor**: cores para as linhas
- **print.summary**: se verdadeiro, o número ótimo de clusters é exibido na tela.

1.4 Boruta

Definição

Boruta é um pacote do R usado para determinar o grau de importância de uma variável. Em um modelo de predição selecionar as variáveis mais importantes ajuda a maximizar a precisão do modelo.

O Boruta inicia com um algoritmo baseado no *Random Forest.* Usando somente o *Random Forest* não é possível calcular o nível de importância de uma variável, já que seu Z score não está diretamente relacionado com a significância estatística da importância da variável. Para evitar isso o Boruta faz o *Random Forest* tanto para os atributos originais quanto para os gerados no processo, como o processo depende da permutação das cópias, a permutação é realizada várias vezes para conseguir resultados consistentes.

[7] Método da interpretação e validação da consistência de clusters de dados. Esta técnica fornece uma sucinta representação gráfica de quão bem cada objeto está disposto no cluster. O valor da silhueta é uma medida de quão similar um objeto é de seu cluster comparado a outros clusters.

Implementação

1 - Primeiro são duplicadas as variáveis, no mínimo deve haver cinco cópias;

2 - Essas cópias são embaralhadas para remover qualquer correlação que venha surgir com a variável alvo;

3 - Combina as variáveis originais com as cópias;

4 - Realiza uma classificação com base no algoritmo de *Random Forest* e faz uma medição da importância de cada variável;

5 - É calculado o Z score: $z = \frac{x - \mu}{\sigma}$ Onde: μ é a média populacional e σ é o desvio padrão populacional e x é a variável aleatória;

6 - Encontra-se o maior Z score entre eles MZSA *(maximum Z score among shadow attributes)*;

7 - Classifica se as variáveis não são importantes se possuírem um Z escore muito menor que o MZSA, então são removidas do processo;

8 - Classifica as variáveis como importante se possuírem um Z score significantemente maior que o MSZA;

9 - Realiza os passos acima até que atinja um número predeterminado de iterações ou até que todas as variáveis estejam marcadas como importante ou não importante.

Por que Boruta?

É um método melhorado do *Random Forest* na parte de mensurar o grau de importância dos atributos. Possui um alto grau de precisão das predições mesmo para relações complexas multivariadas e funciona bem tanto para regressão quanto para classificação.

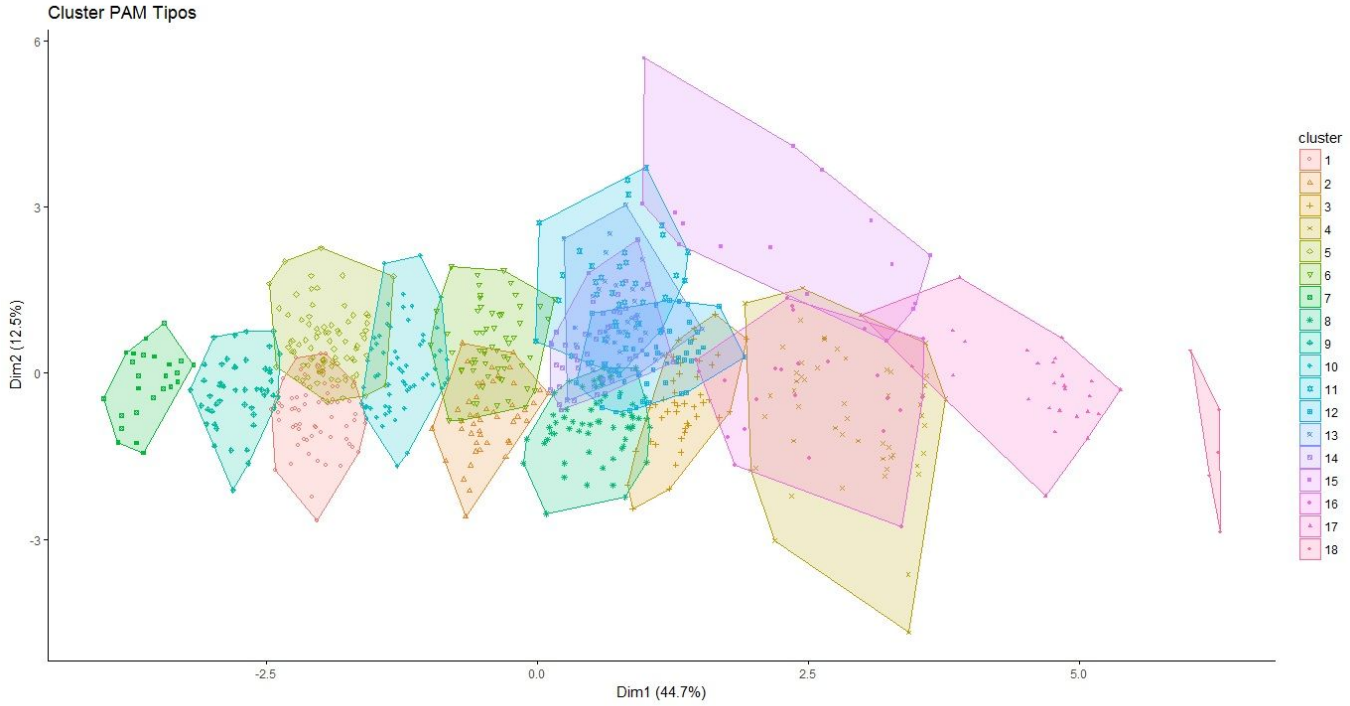
2. RESULTADOS E DISCUSSÃO

Nesta seção, apresentamos gráficos para exibir os resultados alcançados. Todos os gráficos apresentados foram feitos usando o Rstudio, com exceção do gráfico que mostra a taxa de acerto do modelo que será exibido mais adiante. Também discutimos aqui a trajetória tomada pela equipe no desenvolvimento do projeto.

Nossa ideia original foi de identificar os Pokémon, classificando-os de acordo com sua geração. Infelizmente, em nossa primeira análise - documentada no primeiro relatório apresentado à disciplina - foi possível perceber que não havia muitas diferenças entre as gerações, se tratando dos demais atributos.

Portanto, partimos para um novo caminho. Ao invés de classificarmos os Pokémon por gerações, tentaríamos agora classificá-los de acordo com seus tipo. O método utilizado para tal abordagem foi da separação das observações em clusters, em especial através do PAM, como já foi explicado em seções anteriores.

Gráfico 1 : Clusterização com 18 Grupos
Gráfico gerado usando a função PAM



O número ideal de *k* foi definido como 18, visto que, no banco de dados, todos os Pokémon estão distribuídos em exatos 18 tipos. Infelizmente, pela segunda vez, os resultados não foram agradáveis.

Ao analisarmos de perto as divisões dos clusters, pudemos perceber facilmente que as divisões não seguiam nenhum padrão referente aos tipos de Pokémon. Dessa forma, Pokémon de tipos distintos eram agrupados em um mesmo clusters, enquanto que Pokémon de mesmo tipos muitas vezes apareciam separados.

Essa clusterização problemática nos levou a pensar se o número 18 seria realmente o número ideal de clusters. Isso porque talvez alguns tipos pudessem ser tão parecidos que acabariam se “fundindo” no processo. Para resolvermos esse problema, aplicamos o método *silhouette* juntamente com o método do cotovelo para determinarmos o número ótimo de clusters.

Gráfico 2: Número Ótimo de Clusters

O gráfico foi gerado através do método ‘silhouette’ e com o método do cotovelo determinou-se como 2 o número ótimo de clusters para o banco de dados estudado.

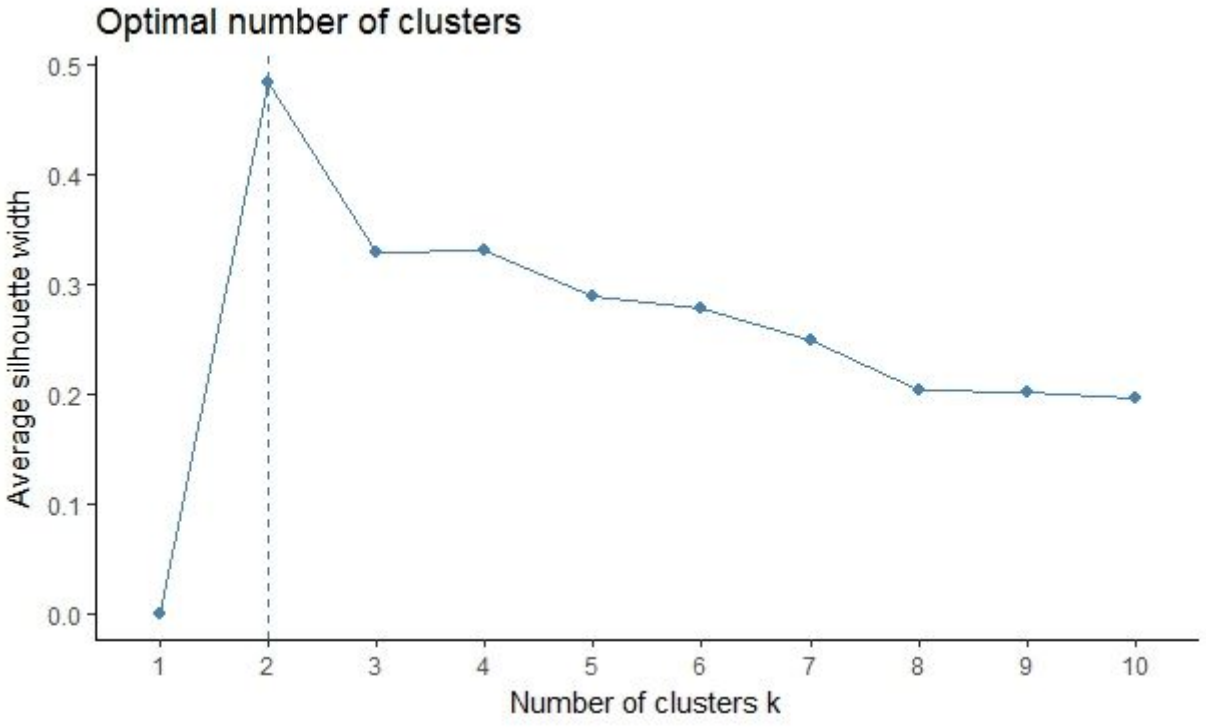


Gráfico 3: Clusterização com 2 grupos Incluindo a Variável ‘Total’

O gráfico foi montado com k = 2 que foi determinado como ideal através do método do cotovelo. O intuito foi descobrir com que padrão o modelo estava separando as ocorrências do banco de dados, como não sabíamos se o atributo ‘Total’ (soma de todos os outros atributos) atrapalhava no modelo então decidimos fazer para os dois casos (com e sem o atributo ‘Total’).

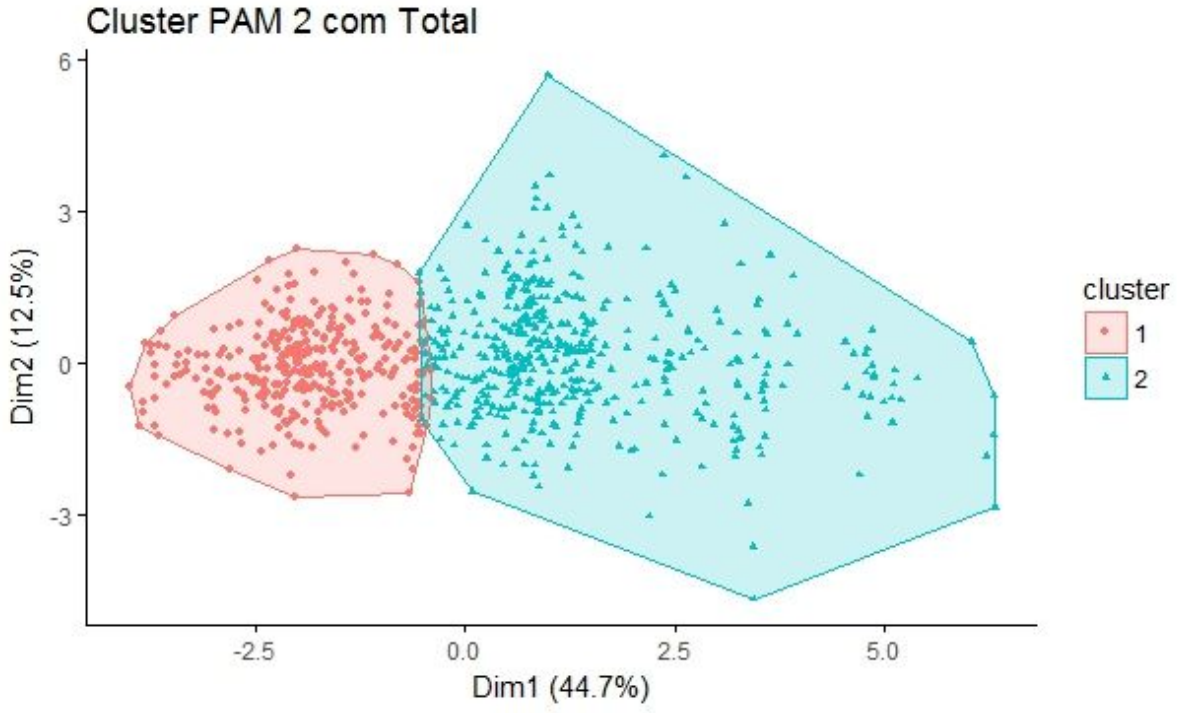
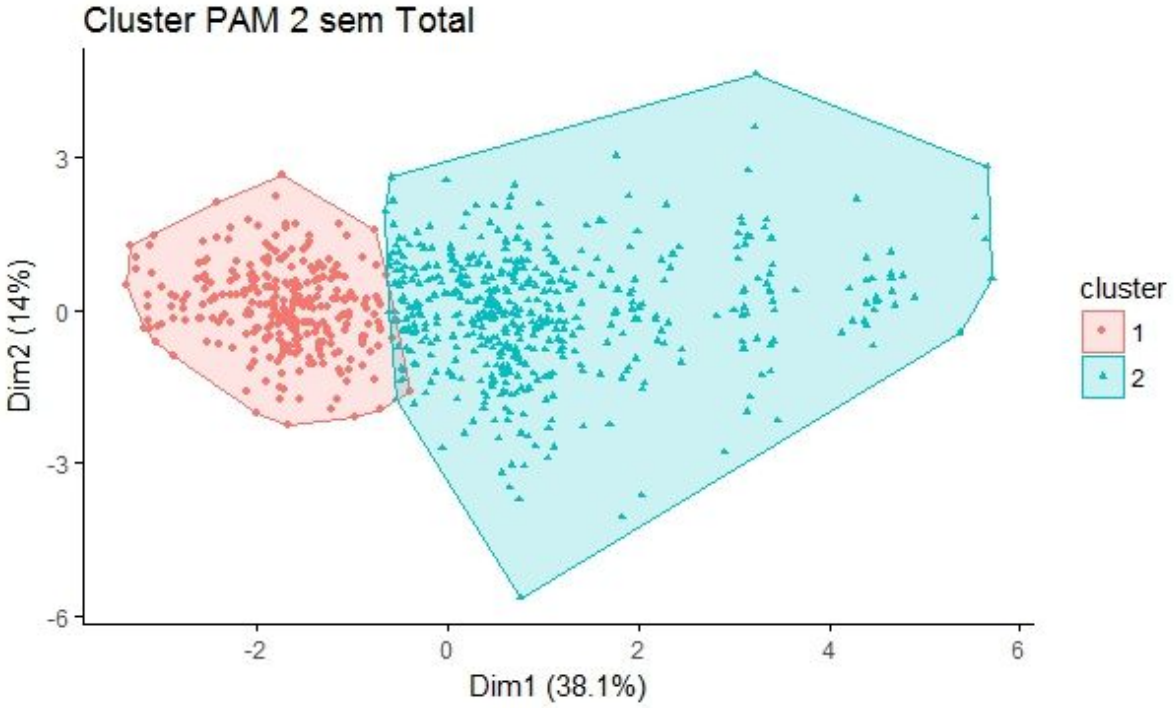


Gráfico 4: Clusterização com 2 Grupos Excluindo a Variável 'Total'
O mesmo caso que o gráfico 3 mas excluindo a coluna 'Total'.



Observando os resultados obtidos, foi possível perceber uma divisão mais sensata, o que nos animou em relação ao resultado. Com dois grupos distintos, tornou-se claro que os clusters dividiam os Pokémon em dois grupos de acordo com sua força - ou poder.

Discutindo, chegamos a conclusão que talvez fosse interessante não mais classificar os Pokémon pelos tipos, mas de acordo com seu poder. Sabíamos que o sistema de evoluções dos Pokémon seria uma boa medida oficial para essa divisão, porém, essa informação não estava presente no nosso banco de dados.

Utilizamos, então, uma tabela de evoluções dos mesmos autores do nosso banco de dados para atualizar manualmente as 800 observações com uma nova variável: Evolução (*Evolution*). Esse atributo consiste de uma variável quantitativa discreta, que abrange valores inteiros entre 1 e 4. De forma que, o número 1 representa as evoluções primárias, o número 2, as secundárias, e o número 3, as ternárias. Além disso, os Pokémon considerados "especiais",

mais fortes que os comuns, como é o caso dos Pokémon lendários e Mega, foram todos considerados de Evolução 4.

Testamos a clusterização com $k = 3$ e $k = 4$, sendo que, quando $k = 3$, os Pokémon lendários e Mega estão agrupados juntos às evoluções ternárias, no grupo de Evolução 3. Veja abaixo os resultados da clusterização com os parâmetros definidos:

Gráfico 5: Clusterização com 3 Grupos Incluindo a Variável 'Total'

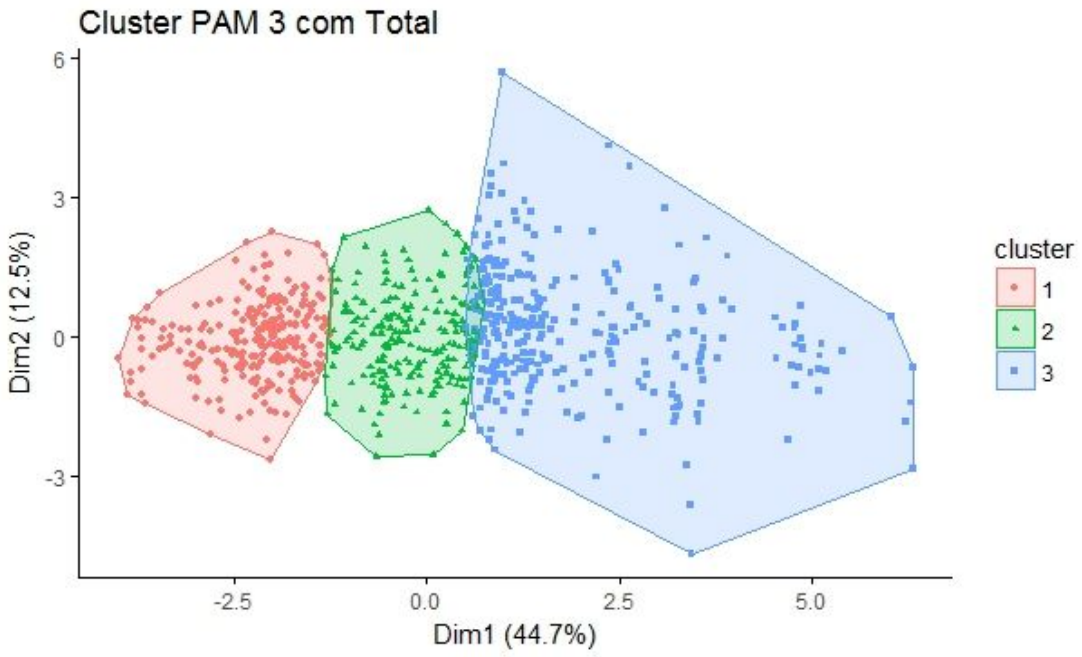


Gráfico 6: Clusterização com 3 Grupos Excluindo a Variável 'Total'

Mesmo caso que o gráfico 5 mas excluindo a coluna 'Total' do modelo.

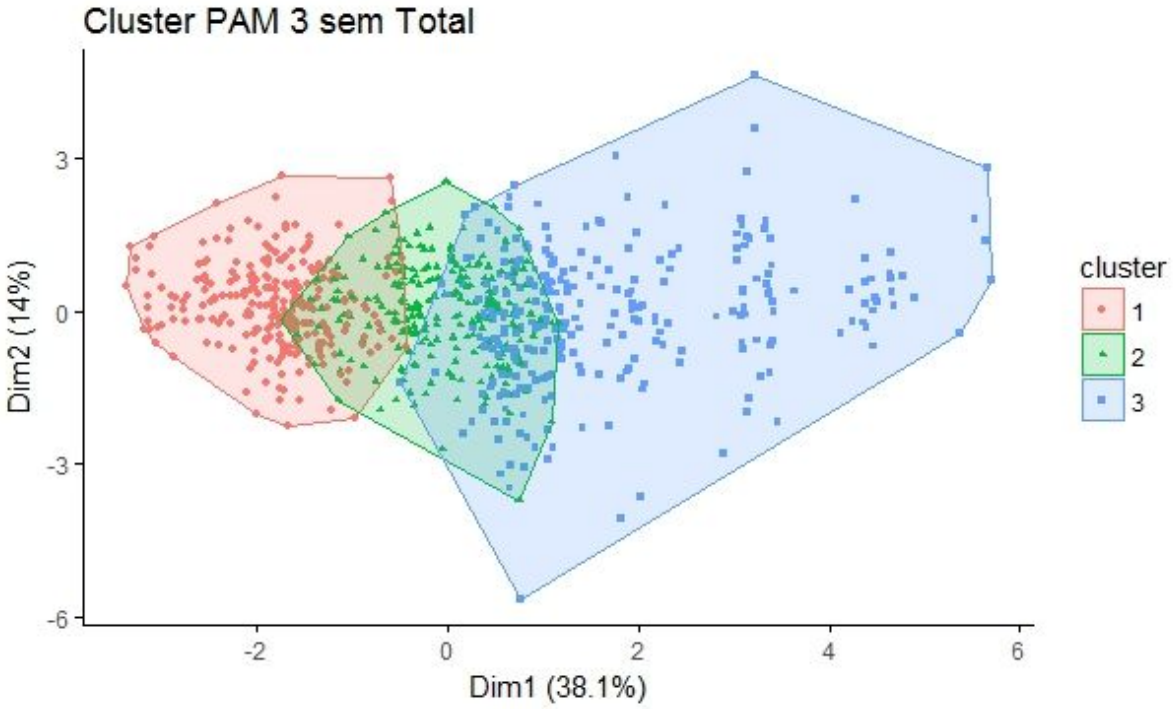
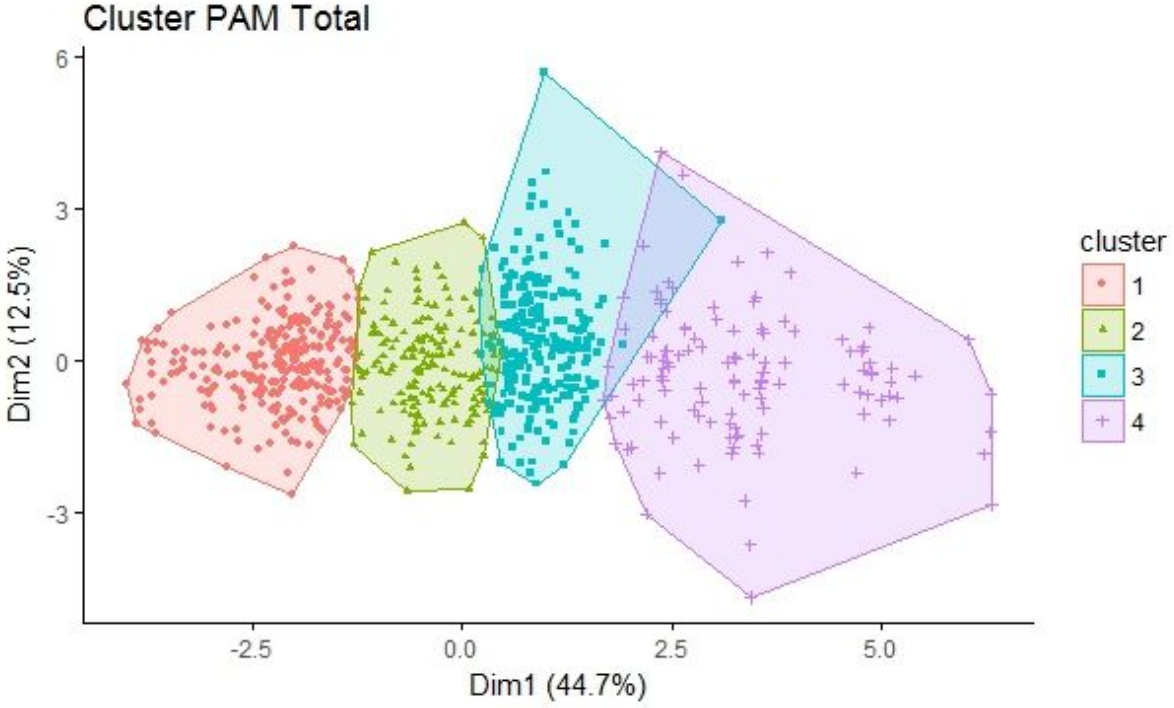


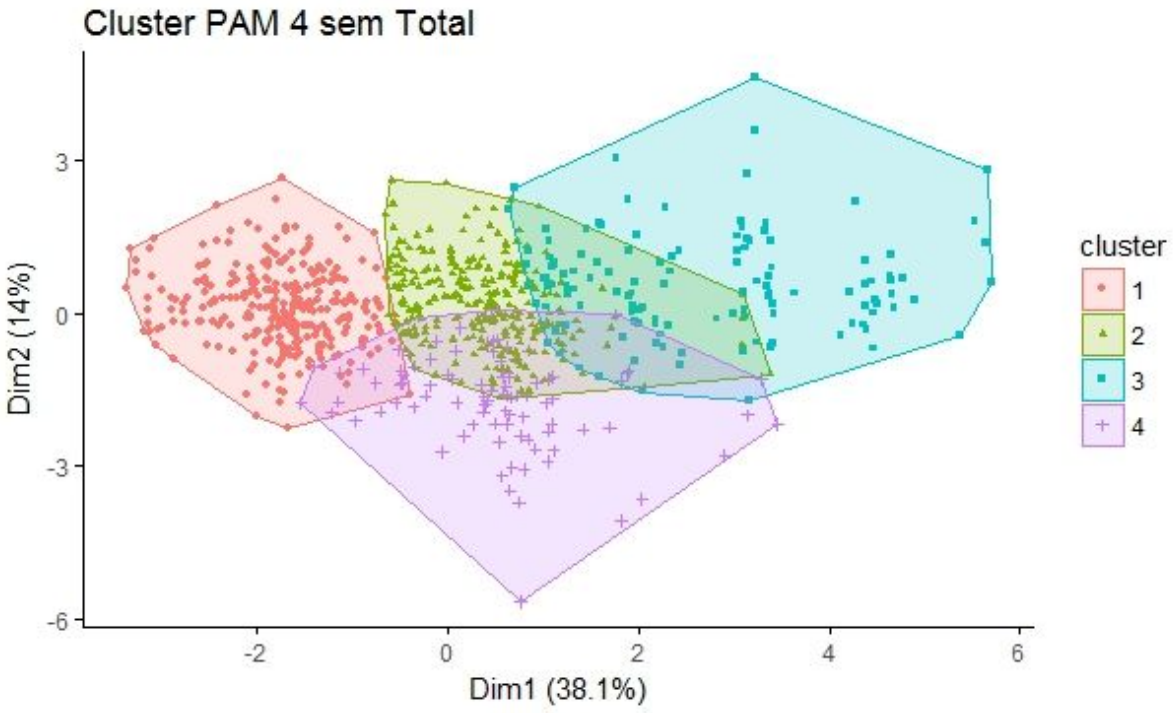
Gráfico 7: Clusterização com 4 Grupos Incluindo a Variável 'Total'



.

Gráfico 8: Clusterização com 4 Grupos Excluindo a Variável 'Total'

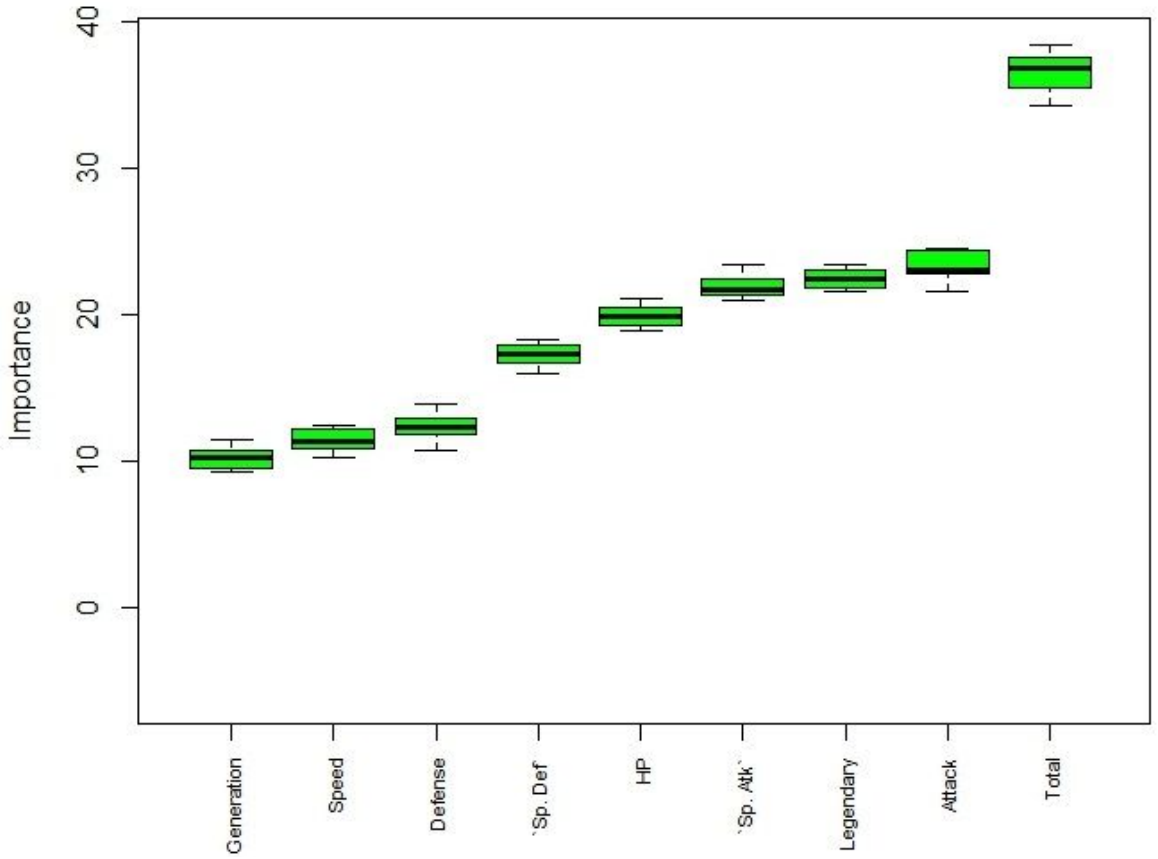
Mesmo caso do gráfico 7 mas sem a variável 'Total'.



Como última medida, na tentativa de melhorar o modelo para a clusterização, resolvemos procurar algum método que nos permitisse descobrir quais variáveis teriam mais importância. Com isso, encontramos o método Boruta, que classifica a importância de cada variável em relação a uma outra - o método já foi explicado em seções anteriores.

Gráfico 9 : Grau de Importância das Variáveis em Relação à Evolução

Este gráfico mostra o grau de importância das variáveis para determinar o patamar de evolução do Pókemon, a função utilizada no R realiza o algoritmo Boruta.



Com a classificação das importâncias de cada variável, testamos o modelo para cada subconjunto de variáveis mais importantes, ou seja, removendo as variáveis menos importantes até que sobrasse apenas o Total, que é a variável mais importante.

Gráfico 10: Taxa de Acerto por Conjunto de Atributos

Este gráfico foi gerado no Excel e mostra a taxa de acerto do modelo conforme as colunas retiradas do modelo. Onde a abscissa representa o número de atributos menos significativos que foram tirados, sendo que em 0 nenhum atributo foi tirado e 8, os 8 atributos menos significativos foram retirados.

A partir dos resultados, pudemos perceber que o melhor momento do modelo é quando ele atinge a taxa de acerto de 77%, que ocorre quando removemos as variáveis “Geração”, “Velocidade” e “Defesa” do classificador. Aplicamos as nossas últimas análises, então, nesse modelo, expostas a seguir.

Gráfico 11: Clusterização com 3 Grupos Excluindo as Variáveis 'Geração', 'Velocidade' e 'Defesa'.

Para maximizar a precisão do modelo utilizamos (conforme o resultado do Gráfico 10) 3 medóides e retiramos as três variáveis de menor importância: defesa, velocidade e geração.

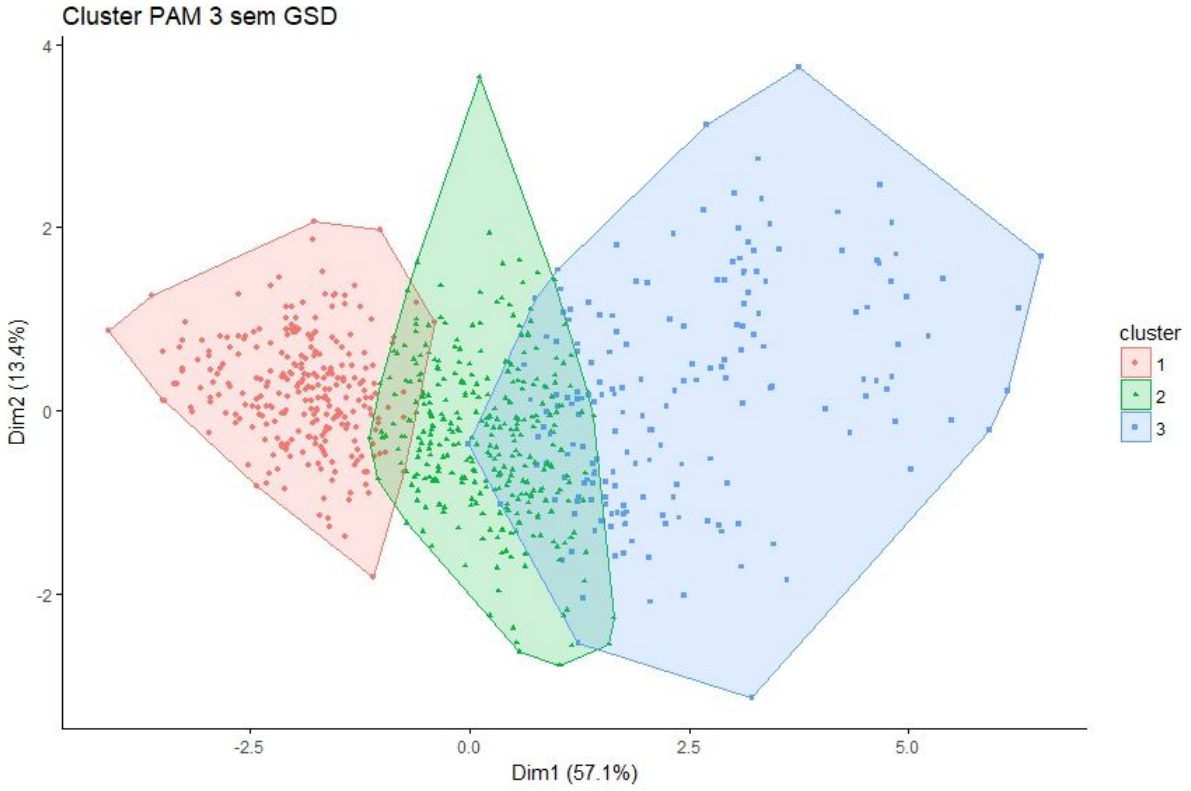


Tabela 1: Matriz de Confusão

Para verificar a precisão da clusterização foi usada uma matriz de confusão feita no Excel. Nas colunas é indicado o valor previsto pelo modelo e nas linhas o valor correto esperado, então na diagonal principal (cor alaranjada) temos as previsões corretas que ele fez. As porcentagens indicam a quantidade de acerto para cada cluster.

		PREVISTO				
		1	**2**	**3**		
ESPERADO	**1**	246	70	13	329	74.77%
	2	31	214	26	271	78.97%
	3	0	44	156	200	78.00%
		277	328	195	800	
		88.81%	65.24%	80.00%		

Tabela 2:Tabela de Verdadeiros Positivos e Negativos

Para facilitar visualizar os resultados geramos uma tabela de falsos positivos e negativos (previsões incorretas) e verdadeiros positivos e negativos (previsões corretas) para cada cluster.

	VP	VN	FP	FN
1	246	370	31	83
2	214	402	114	57
3	156	460	39	44

3. CONCLUSÃO

Com as análises realizadas, foi possível perceber que não há distinção considerável entre os Pokémon se tratarmos de suas gerações ou até mesmo de seus tipos, mas que uma distinção bastante específica pode ser notada em relação ao seu grau de evolução.

Também foi possível notar que os Pokémon lendários estão sempre entre os Pokémon mais fortes, bem como os Mega. Isso pode ser verificado pelo fato de que 100% dos Pokémon lendários ficaram nos grupos de mais alta ordem na clusterização (com $k = 3$ e $k = 4$) e 92% dos Mega ficaram no cluster 3, quando $k = 3$, de forma que apenas quatro Pokémon Mega não ficaram contidos nesse grupo.

É importante notarmos que as variáveis “Geração”, “Velocidade” e “Defesa” não são tão relevantes para determinar a evolução de um Pokémon, de forma que o modelo fica com uma melhor taxa de acerto quando estas variáveis são retiradas.

É interessante notar que, mesmo com erros nos agrupamentos, não há inversões de evoluções em uma mesma família de Pokémon. Ou seja, o

método erra considerando que um Pokémon de evolução 2 esteja no mesmo grupo que seu antecessor (de evolução 1), mas nunca coloca um Pokémon mais fraco em um grupo mais forte e ao mesmo tempo um Pokémon mais forte, da mesma família, em um grupo mais fraco que o primeiro.

Em suma, apesar das reviravoltas, ficamos felizes com o resultado final, considerando uma taxa de acerto de 77% como uma ótima medida, tendo em vista os esforços e métodos utilizados.

4. REFERÊNCIAS BIBLIOGRÁFICAS

List of Pokémon by Evolution Family. Bulbapedia.

BHALLA, Deepanshu; Select Important Variables using Boruta Algorithm. Data Science Central, 2017.

DUTTA, Debaratti. How to perform feature selection (i.e. pick important variables) using Boruta Package in R? Analytics Vidhya, 2016.

HAN, Jiawei; PEI, Jian; KAMBER, Micheline. Data mining: concepts and techniques. Elsevier, 2011.

KASSAMBARA, Alboukadel; fviz_nbclust: Dertermining And Visualizing The Optimal Number Of Clusters. R Documentation.

KURSA, Miron; Boruta: Wrapper Algorithm for All Relevant Feature Selection. R Documentation.

MACQUEEN, James et al. Some methods for classification and analysis of multivariate observations. In: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability. 1967. p. 281-297.

MAECHLER, Martin; pam: Partitioning Around Medoids. R Documentation.

R-core; kmeans: K-Means Clustering. R Documentation.

“Fuzzy clustering”, “K-Means clustering” e “Vector quantization”. Origem: Wikipedia.