Relatório Técnico: K-means no dataset Human Activity Recognition Using Smartphones

Autores: Aurélio José Ribeiro da Silva, Andressa Luíza Costa de Carvalho

SUMÁRIO

RESUMO

Este projeto tem como objetivo aplicar e avaliar o algoritmo de K-means no dataset *Human Activity Recognition Using Smartphones*, utilizando dados coletados de sensores de smartphones. A análise envolve desde a exploração inicial dos dados, passando pela seleção de variáveis relevantes, normalização, redução de dimensionalidade, até a definição do número ideal de clusters. Métricas como Inércia e *Silhouette Score* foram utilizadas para validar o modelo. Resultados promissores foram obtidos com K=3 clusters, mostrando a capacidade de separar as atividades humanas em grupos distintos.

INTRODUÇÃO

O reconhecimento de atividades humanas (*Human Activity Recognition* - HAR) tem se tornado uma área de pesquisa de crescente importância devido ao avanço de dispositivos móveis equipados com sensores embarcados, como smartphones. A capacidade de identificar atividades físicas a partir de dados sensoriais oferece inúmeras aplicações em áreas como saúde, *fitness*, monitoramento de idosos e reabilitação. No contexto do HAR, sensores como acelerômetros e giroscópios têm sido amplamente utilizados para coletar dados sobre os movimentos e a posição do corpo, proporcionando informações valiosas sobre a execução de atividades físicas, como caminhar, correr e subir escadas.

O estudo realizado por *Anguita et al.* apresenta uma abordagem inovadora ao utilizar smartphones equipados com esses sensores para coletar dados de atividades humanas. O trabalho resultou na criação de um conjunto de dados público e abrangente, amplamente utilizado na comunidade científica, e demonstrou resultados promissores ao empregar técnicas de aprendizado de máquina para a classificação e agrupamento das atividades.

Entre as diversas abordagens para HAR, o *K-means* destaca-se como um método eficiente para agrupamento de dados em cenários onde não se tem rótulos explícitos para as atividades, ou seja, para descobrir padrões em dados não rotulados. O *K-means* é um algoritmo de clustering que agrupa as observações em clusters baseados em características similares, sendo particularmente eficaz quando se lida com dados sensoriais multidimensionais, como os coletados por smartphones.

Uma das principais limitações enfrentadas em sistemas de reconhecimento de atividades é a alta dimensionalidade dos dados gerados pelos sensores, o que pode resultar em alto custo computacional e redundância de informações. Para mitigar esses desafios, este trabalho emprega a Análise de Componentes Principais (PCA) como técnica de redução de dimensionalidade. O PCA transforma os dados originais em um conjunto de componentes principais, preservando a maior parte da variância nos dados e eliminando redundâncias. Essa abordagem visa melhorar a eficiência computacional do *K-means*, reduzir o risco de *overfitting* e melhorar a capacidade de generalização do modelo.

CONJUNTO DE DADOS

O conjunto de dados utilizado neste projeto foi obtido do experimento *Human Activity Recognition Using Smartphones*, que contou com a participação de 30 voluntários com idades entre 19 e 48 anos. Cada participante realizou seis atividades distintas enquanto utilizava um smartphone (*Samsung Galaxy S II*) preso à cintura. As atividades monitoradas foram:

- **Andar** (*WALKING*)
- **Subir escadas** (*WALKING_UPSTAIRS*)
- **Descer escadas** (*WALKING_DOWNSTAIRS*)
- **Sentar** (*SITTING*)
- **Ficar em pé** (*STANDING*)
- **Deitar** (*LAYING*)

Características do Conjunto de Dados

- Número total de amostras: 10.299
- Número de variáveis (features): 561, extraídas dos domínios do tempo e da frequência
- Taxa de amostragem: 50 Hz (50 leituras por segundo)
- Divisão dos dados:
    - 70% dos voluntários foram selecionados aleatoriamente para o conjunto de treino
    - 30% dos voluntários foram utilizados no conjunto de teste

Aquisição e Processamento dos Dados

Durante o experimento, os dados foram coletados por dois sensores embutidos em um smartphone, sendo eles um acelerômetro e um giroscópio. O acelerômetro capturou a aceleração linear tridimensional nos eixos X, Y e Z, enquanto o giroscópio registrou a velocidade angular tridimensional nos mesmos eixos.

Após a coleta, os sinais passaram por um processo de pré-processamento, no qual foram aplicados filtros de remoção de ruído para garantir maior precisão nos dados. Em seguida, os sinais foram segmentados em janelas temporais de 2,56 segundos, com uma sobreposição de 50% entre as janelas consecutivas, resultando em 128 leituras por janela.

A aceleração medida foi então decomposta em dois componentes distintos:

- Aceleração do corpo, que reflete o movimento gerado pelas atividades realizadas pelo voluntário.

- Gravidade, que representa o componente de baixa frequência do sinal, isolado por meio de um filtro Butterworth com frequência de corte de 0,3 Hz.

Extração de Atributos (Features)

Para cada janela segmentada, foram extraídos 561 atributos que descrevem as características dos sinais nos domínios do tempo e da frequência. Entre as estatísticas calculadas estão:

- Média, desvio padrão, valor máximo, valor mínimo e energia dos sinais.
- Correlação entre os eixos triaxiais, que fornece informações sobre a relação entre os movimentos nos diferentes eixos.
- Frequências dominantes e entropia, que refletem a distribuição e a complexidade dos sinais no domínio da frequência.

Cada amostra do conjunto de dados final contém:

- A aceleração total triaxial registrada pelos sensores.
- A aceleração estimada do corpo.
- A velocidade angular triaxial.
- Um vetor com os 561 atributos extraídos.
- O rótulo da atividade realizada.
- O identificador do voluntário que executou a atividade correspondente.

METODOLOGIA

Análise

Na parte de análise exploratória no *Google Colab*, que é uma etapa inicial do estudo, procuramos compreender a natureza dos dados, identificando características, padrões, qualidade dos dados e obter os primeiros insights sobre as distribuições dos dados.

Descrição do Conjunto de Dados

As variáveis representam diferentes estatísticas calculadas a partir dos sinais dos sensores, como média, desvio padrão e coeficiente de correlação entre eixos X, Y e Z. Com isso temos um dataset de 10299 entradas e 561 colunas, os valores são do tipo float64 e tem um uso de memória de 44.1MB.

```
print(X.shape) #Verificando tamanho do dataset

(10299, 561)

print(X.info()) #Verificando estrutura do dataset

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 10299 entries, 0 to 10298

Columns: 561 entries, tBodyAcc-mean()-X to angle(Z,gravityMean)

dtypes: float64(561)

memory usage: 44.1 MB

None
```

Resumo Estatístico

Estatísticas descritivas foram geradas para entender a distribuição das variáveis, detectando outliers e padrões globais, abaixo um recorte das primeiras 3 colunas descritas. Não foi possível anexar todos devido as dimensões dos dados, exemplificamos apenas as primeiras colunas.

```
# Estatísticas descritivas

print(X.describe())
```

	tBodyAcc-mean()-X	tBodyAcc-mean()-Y	tBodyAcc-mean()-Z
count	10299.000000	10299.000000	10299.000000
mean	0.274347	-0.017743	-0.108925
std	0.067628	0.037128	0.053033
min	-1.000000	-1.000000	-1.000000
25%	0.262625	-0.024902	-0.121019
50%	0.277174	-0.017162	-0.108596
75%	0.288354	-0.010625	-0.097589
max	1.000000	1.000000	1.000000

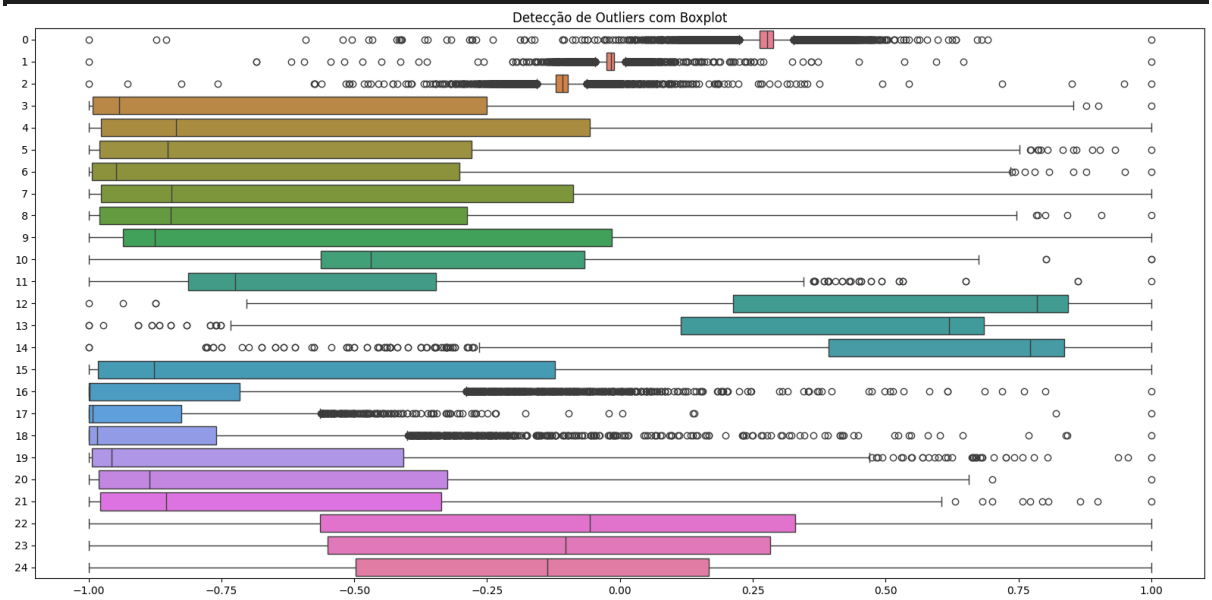
Tratamento de Dados Faltantes, Anomalias e Ajustes

Não foram identificados valores faltantes no conjunto de dados, mas foram observados outliers em algumas variáveis, tratados com normalização, e mantidos para verificar a eficácia do algoritmo mesmo com esses dados.

```
print('Verificar dados faltantes: ' + str(X.isnull().any().any()))
```

```
Verificar dados faltantes: False
```

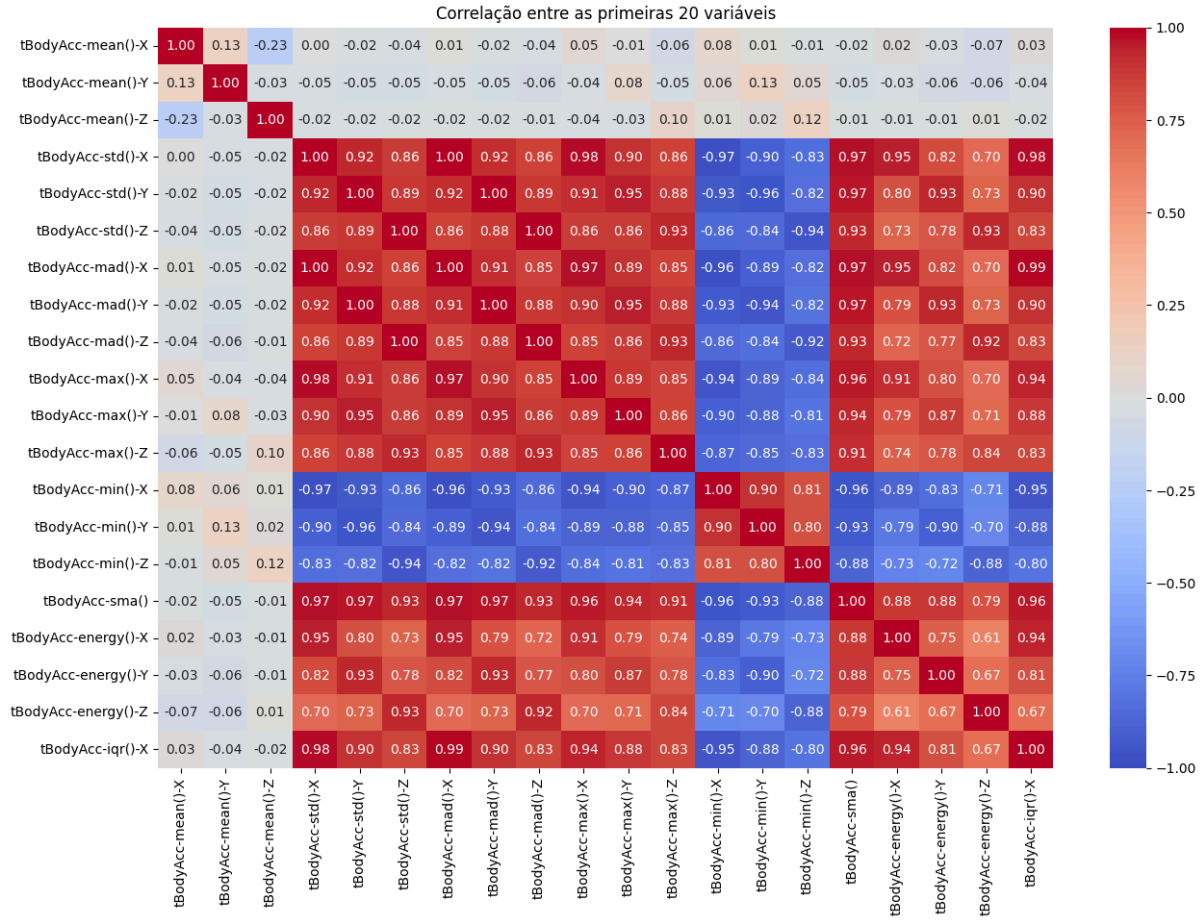
```
plt.figure(figsize=(16, 8))

sns.boxplot(data=X.iloc[:,:25], orient='h')

plt.title("Detecção de Outliers com Boxplot")

plt.tight_layout()

plt.show()
```



Correlação

Uma análise de correlação mostrou fortes relações (> 0,9) entre variáveis associadas a movimentos similares. Foi possível observar 94 variáveis com fortes relações, o que foi algo interessante e mostrou que os dados possuem relação entre si. Foi feito com os seguintes códigos para geração da matriz e das contagem das variáveis relacionadas respectivamente.

```
correlation_matrix = X.iloc[:, :20].corr()

correlation_df = correlation_matrix.reset_index()

plt.figure(figsize=(15, 10))

sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap='coolwarm',
vmin=-1, vmax=1, cbar=True)

plt.title('Correlação entre as primeiras 20 variáveis')

plt.show()
```



Depois, fizemos uma seleção das variáveis que mais se correlacionam (acima de 90%), exceto pelos valores da diagonal principal da matriz, das 20 primeiras features selecionadas para a análise da correlação.

```
high_corr_pairs
correlation_matrix.unstack().sort_values(ascending=False)

high_corr_pairs = high_corr_pairs[high_corr_pairs > 0.9]

high_corr_pairs.iloc[20:]
```

tBodyAcc-mad()-X	tBodyAcc-std()-X	0.998662
tBodyAcc-std()-X	tBodyAcc-mad()-X	0.998662
tBodyAcc-mad()-Y	tBodyAcc-std()-Y	0.997510
tBodyAcc-std()-Y	tBodyAcc-mad()-Y	0.997510
tBodyAcc-mad()-Z	tBodyAcc-std()-Z	0.997360
...	...	...
tBodyAcc-sma()	tBodyAcc-max()-Z	0.910813
tBodyAcc-max()-X	tBodyAcc-mad()-Y	0.904283
tBodyAcc-mad()-Y	tBodyAcc-max()-X	0.904283
tBodyAcc-std()-Y	tBodyAcc-iqr()-X	0.903326
tBodyAcc-iqr()-X	tBodyAcc-std()-Y	0.903326

74 rows × 1 columns

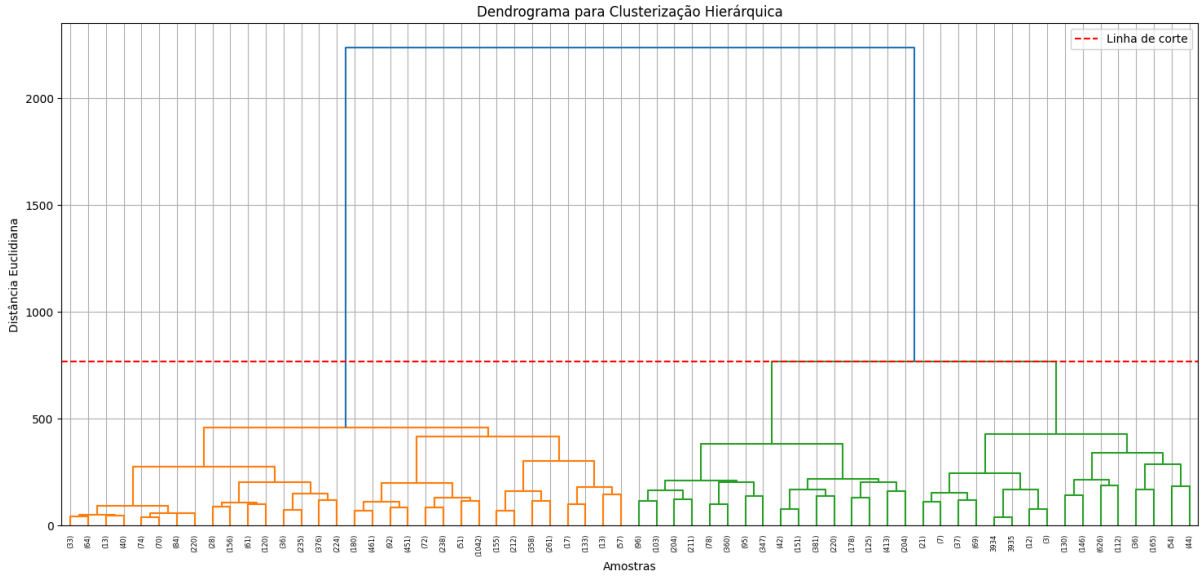
dtype: float64

Dendograma

O dendrograma foi gerado para explorar a hierarquia dos agrupamentos e sugerir o número adequado de clusters antes da aplicação do K-means. O método de *linkage* utilizado foi o **Ward**, que minimiza a variância dentro dos clusters:

```
scaler = StandardScaler()

X_scaled = scaler.fit_transform(X)

linkage_matrix = linkage(X_scaled, method='ward')

distances = linkage_matrix[:, 2]

sorted_distances = np.sort(distances)

differences = np.diff(sorted_distances)

max_diff_index = np.argmax(differences)

plt.figure(figsize=(18, 8))
```

```
dendrogram(linkage_matrix, truncate_mode='level', p=5)

plt.axhline(y=sorted_distances[max_diff_index], color='r',
linestyle='--', label='Linha de corte')

plt.title('Dendrograma para Clusterização Hierárquica')

plt.xlabel('Amostras')

plt.ylabel('Distância Euclidiana')

plt.legend()

plt.grid(True)

plt.show()
```




```
print(f"Maior salto em distância entre os clusters:
{differences[max_diff_index]}")

print(f"Distância ideal para corte:
{sorted_distances[max_diff_index]}")
```

```
Maior salto em distância entre os clusters: 1467.8937398800451

Distância ideal para corte: 768.6600669928569
```

```
cut_distance = 768.66

clusters = fcluster(linkage_matrix, cut_distance, criterion='distance')

num_clusters = len(set(clusters))
```

```
print(f"Número de clusters sugeridos: {num_clusters}")

Número de clusters sugeridos: 3
```

O maior “salto” na vertical entre dois níveis consecutivos de fusão representa o ponto em que unir dois clusters exige um aumento significativo na distância e isso sugere que eles são melhor mantidos separados. A partir disso pegamos o número ideal de cluster com base nessas distâncias com a função *fcluster* que conta os clusters a partir da distância, **o número de clusters sugerido foi 3**.

Pré-processamento dos dados

Normalização

Os dados foram normalizados usando *StandardScaler* para garantir que todas as variáveis tivessem a mesma importância durante o agrupamento e que os outliers não direcionassem os resultados.

```
scaler = StandardScaler()

X_scaled = scaler.fit_transform(X)
```

PCA

```
pca = PCA(n_components=3)

X_scaled_pca = pca.fit_transform(X_scaled)

print(f'Explained variance ratio: {pca.explained_variance_ratio_}')

Explained variance ratio: [0.50738221 0.06239186 0.02692564]
```

Aqui representamos a proporção de variância total dos dados explicadas por cada componente principal selecionado. Cerca de 60% da variância total dos dados é explicada pelos 3 primeiros componentes principais. Foi verificado que a adição de mais componentes não trazia diferenças significativas, não mais do que 1%, sugerindo que há muita redundância nos dados depois dessas primeiras componentes. Como a maior parte da variância está no PC1, isso significa que os dados têm uma estrutura que é fortemente linear. O PCA com 0.95 foi utilizado porém apresentou os mesmos desempenhos que não eram substancialmente diferentes para 3 componentes.

IMPLEMENTAÇÃO DO ALGORITMO

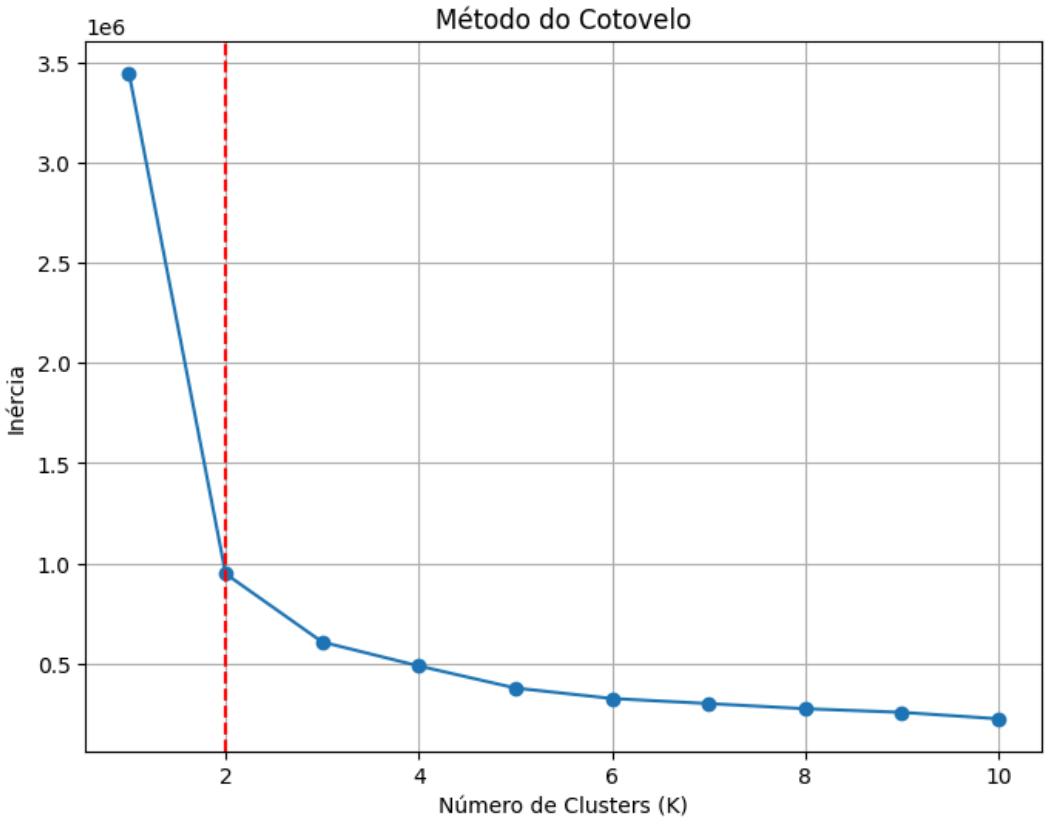
Divisão dos Dados

O *KMeans* é um método de aprendizado não-supervisionado, logo ele não utiliza uma divisão rotulada, ele “aprende” as relações de entrada-saída para realizar as predições buscando padrões ou agrupamentos nos dados de entrada.

Escolha do número de clusters

O método do cotovelo ajuda a encontrar o valor de K onde a inércia começa a diminuir mais lentamente, a inércia por sua vez é uma medida de quão bem os pontos estão agrupados em torno dos centroides.

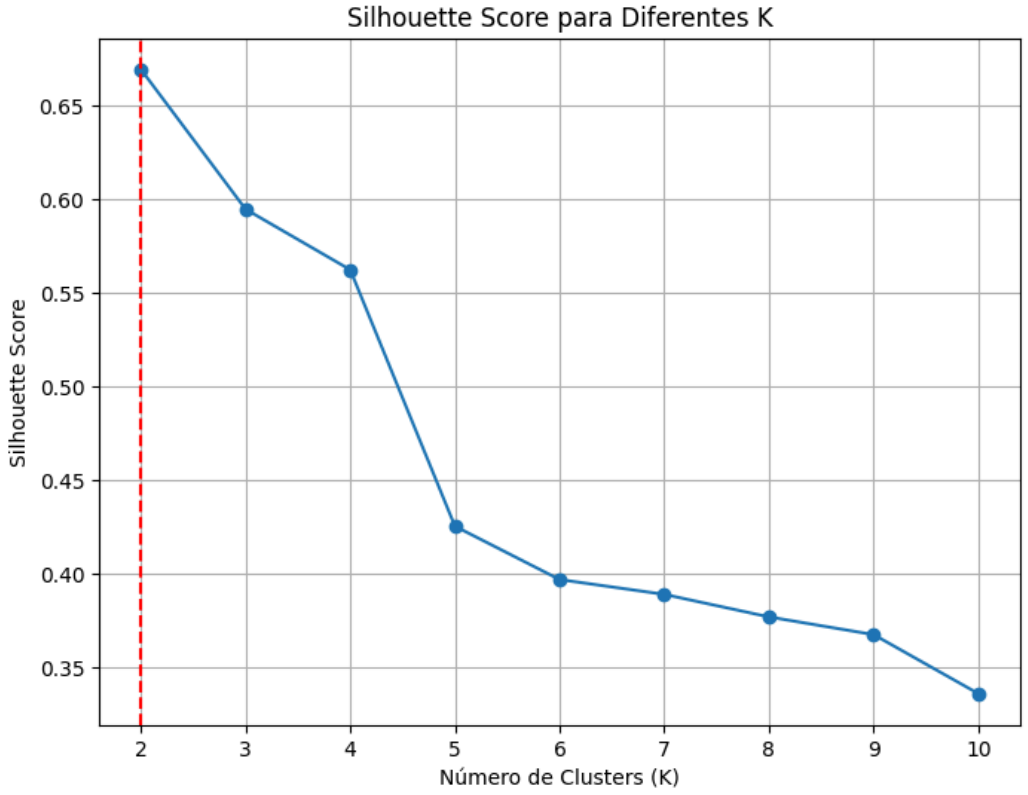
```
inertias = []

K = range(1, 11)

for k in K:

    kmeans = KMeans(n_clusters=k, random_state=42)

    kmeans.fit(X_scaled_pca)

    inertias.append(kmeans.inertia_)

best_k_inertia = K[inertias.index(min(inertias[1:], key=lambda x:
abs(x-inertias[0])))]

plt.figure(figsize=(8, 6))

plt.plot(K, inertias, marker='o')

plt.axvline(x=best_k_inertia, color='r', linestyle='--')

plt.title("Método do Cotovelo")

plt.xlabel("Número de Clusters (K)")

plt.ylabel("Inércia")

plt.grid()

plt.show()

print(f"O melhor valor de K baseado no cotovelo é: {best_k_inertia}")
```



O Silhouette Score avalia a qualidade dos clusters pela coesão dentro deles e a separação entre eles. O valor varia de -1 e 1.

```
silhouette_scores = []

for k in range(2, 11):

    kmeans = KMeans(n_clusters=k, random_state=42)

    clusters = kmeans.fit_predict(X_scaled_pca)

    silhouette_scores.append(silhouette_score(X_scaled_pca, clusters))

best_k_silhouette = range(2,
11)[silhouette_scores.index(max(silhouette_scores))]

plt.figure(figsize=(8, 6))

plt.plot(range(2, 11), silhouette_scores, marker='o')

plt.axvline(x=best_k_silhouette, color='r', linestyle='--')

plt.title("Silhouette Score para Diferentes K")

plt.xlabel("Número de Clusters (K)")
```

```
plt.ylabel("Silhouette Score")

plt.grid()

plt.show()

print(f"O melhor valor de K baseado no Silhouette Score é:
{best_k_silhouette}")
```



Para ambos os métodos o número de clusters escolhido foi 2:

```
O melhor valor de K baseado no cotovelo é: 2
```

```
O melhor valor de K baseado no Silhouette Score é: 2
```

Validação e Ajuste de Hiperparâmetros

O número ideal de clusters foi determinado usando o método do cotovelo e o *Silhouette Score*. Além disso, levando em consideração também o dendograma e com isso escolhemos o K=3, pois, em relação ao 2 ele diminui o valor da inércia e se mantém bem no *Silhouette Score* Médio.

RESULTADOS

Métricas de Avaliação

Inércia Final do Modelo

A *Inércia* é uma métrica que representa a soma das distâncias quadradas entre cada ponto de dados e o centroide mais próximo. Uma inércia menor indica que os dados estão mais próximos dos centroides, sugerindo uma boa coesão dentro dos clusters.

```
n_clusters = 3

kmeans = KMeans(n_clusters=n_clusters, random_state=42)

clusters = kmeans.fit_predict(X_scaled_pca)

centroides = kmeans.cluster_centers_

inercia = kmeans.inertia_

print("Centroides:", centroides)

print("Inércia Final:", inercia)
```

```
Centroides: [[-14.24102101   1.04105343   0.56161118]
 [ 11.84820169  -4.2482188   -1.6340863 ]
 [ 27.29186296   4.95111562   1.32019766]]
Inércia Final: 606821.041936778
```

Essa inércia indica uma boa separação dos clusters, porém, considerando a alta dimensionalidade dos dados, ainda há margem para melhorias na distribuição dos pontos.

Silhouette Score Final

O *Silhouette Score* varia entre -1 e 1, e mede a qualidade do agrupamento baseado na proximidade dos pontos em relação ao centroide, isto é mede a coerência dos dados dentro de cada cluster levando em consideração a proximidade dos pontos em relação ao centroide de seu próprio cluster e a distância média entre os pontos de diferentes clusters.

```
silhouette_avg = silhouette_score(X_scaled_pca, clusters)

print("Silhouette Score Médio:", silhouette_avg)

Silhouette Score Médio: 0.5941863790133066
```

O Silhouette Score Médio obtido foi de 0,594, o que indica que, em geral, os clusters estão bem definidos e separados, mas não de forma perfeitamente clara. Este valor sugere que os pontos de dados estão razoavelmente distantes uns dos outros, mas alguns podem estar nas fronteiras entre os clusters, indicando uma certa sobreposição ou ambiguidade na classificação. Embora o valor de 0,59 seja aceitável, ele indica que há espaço para melhorias.

Consistência Entre Execuções

A consistência entre execuções foi avaliada usando o Adjusted Rand Score (ARI), que mede a similaridade entre os agrupamentos gerados em diferentes execuções do K-means com K=2 e K=3.

```
kmeans1 = KMeans(n_clusters=2, init='k-means++',
random_state=42).fit(X_scaled_pca)

kmeans2 = KMeans(n_clusters=3, init='k-means++',
random_state=43).fit(X_scaled_pca)

print("Consistência entre execuções:",
adjusted_rand_score(kmeans1.labels_, kmeans2.labels_))

Consistência entre execuções: 0.8079447751403952
```

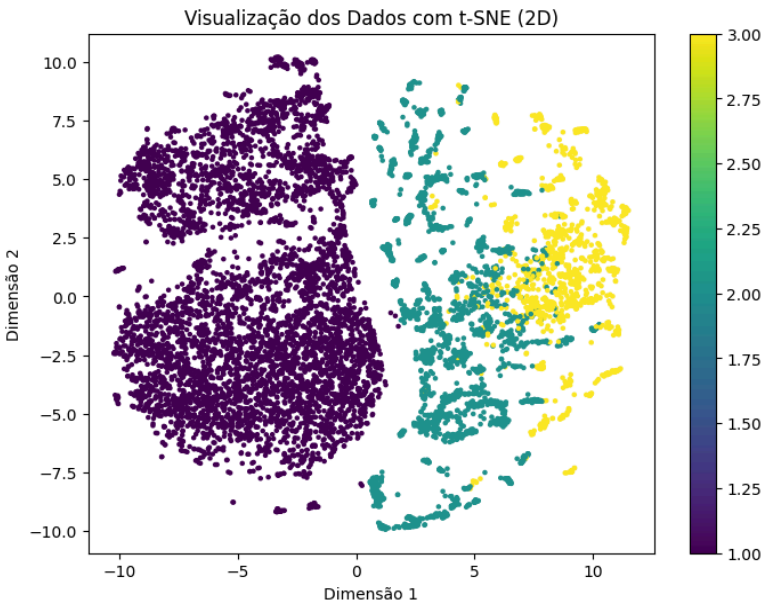
O valor calculado de ARI = 0.8079 indica uma similaridade relativamente alta entre os agrupamentos gerados em diferentes execuções. O fato de o ARI não ter chegado mais próximo de 1 sugere que os dados são sensíveis à escolha do número de clusters. Isso implica que o número de clusters pode afetar consideravelmente a estrutura dos agrupamentos.

VISUALIZAÇÕES

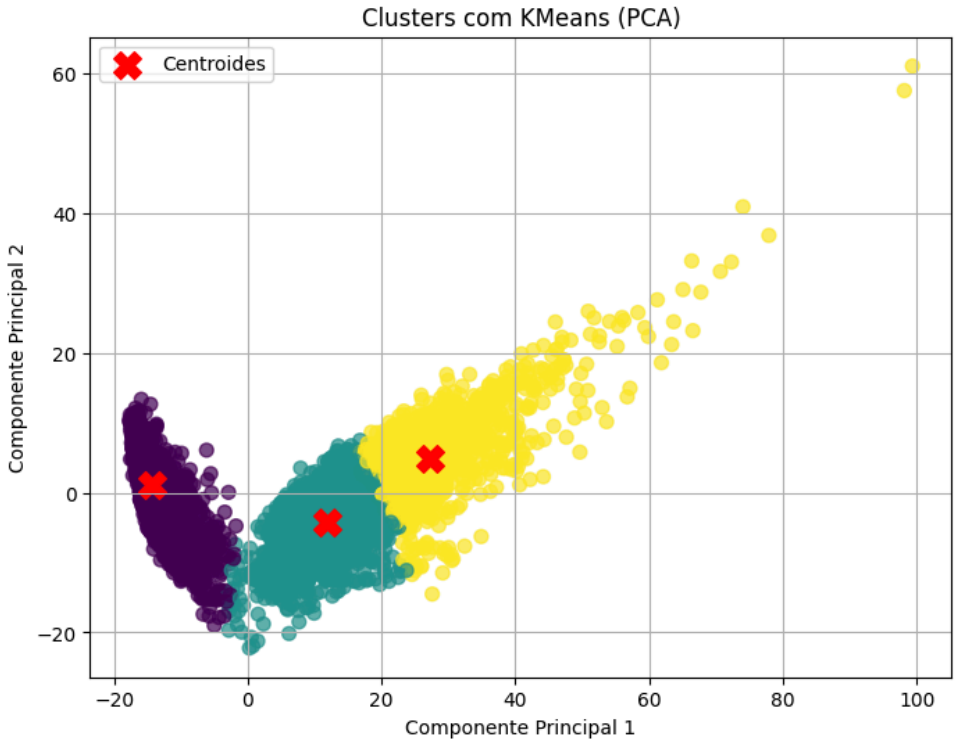
Visualização com t-SNE

A técnica do t-SNE (t-Distributed Stochastic Neighbor Embedding) reduz a dimensionalidade para explorar e visualizar dados de alta dimensão. Ele reduz os dados de alta dimensão para duas dimensões (neste caso) de forma que a relação de proximidade entre os pontos no espaço original seja mantida no espaço reduzido, dessa forma, ele ajuda a identificar padrões, agrupamentos ou tendências nos dados.

```
tsne = TSNE(n_components=2, perplexity=30, max_iter=300,
random_state=42)

X_tsne = tsne.fit_transform(X_scaled)

plt.figure(figsize=(8, 6))

plt.scatter(X_tsne[:, 0], X_tsne[:, 1], c=clusters, cmap='viridis',
s=5)

plt.title('Visualização dos Dados com t-SNE (2D)')

plt.xlabel('Dimensão 1')

plt.ylabel('Dimensão 2')

plt.colorbar()

plt.show()
```



Visualização dos Clusters com PCA



Os marcadores em formato de "X" vermelhos representam os centroides de cada cluster. Os centroides são os pontos médios em torno dos quais os dados de cada cluster estão agrupados.

```
plt.figure(figsize=(8, 6))

plt.scatter(X_scaled_pca[:, 0], X_scaled_pca[:, 1], c=clusters,
cmap='viridis', s=50, alpha=0.7)

plt.scatter(centroides[:, 0], centroides[:, 1], c='red', marker='X',
s=200, label='Centroides')

plt.title("Clusters com KMeans (PCA)")

plt.xlabel("Componente Principal 1")

plt.ylabel("Componente Principal 2")

plt.legend()

plt.grid()

plt.show()
```

DISCUSSÃO

Com base na análise da clusterização hierárquica e na interpretação visual do dendrograma, identificamos que a estrutura dos dados é bem representada por 3 clusters principais, apesar do chute inicial ter sido 6. Essa escolha foi reforçada pela análise visual após a redução dimensional com PCA e t-SNE, que demonstraram uma separação razoável entre os grupos, ainda que os dados do Elbow e Silhouette apontassem 2.

A aplicação do PCA mostrou-se útil na redução da dimensionalidade dos dados, permitindo preservar mais de 50% da variância total com apenas três componentes principais. No entanto, foi observado que a projeção em menor dimensionalidade manteve resultados consistentes com a análise original, indicando que os dados apresentam uma estrutura intrínseca linear ou que os clusters são relativamente bem definidos mesmo em dimensões mais altas.

O algoritmo *K-Means* apresentou boa separação visual dos clusters, especialmente após a redução dimensional. Entretanto, sua dependência da definição prévia do número de clusters pode ter influenciado os resultados. A escolha de 3 clusters se mostrou razoável para representar as atividades humanas registradas no *dataset*, mas essa configuração foi feita com base na análise visual e heurística, o que traz certo grau de subjetividade.

Ainda assim, a análise quantitativa revelou desafios importantes. O valor mediano do *Silhouette Score*, combinado com um alto valor de Inércia, sugere que algumas amostras podem estar localizadas em regiões de transição entre clusters, indicando que elas não estão claramente associadas a um único grupo. Esse comportamento pode ser explicado, em parte, pela alta dimensionalidade inicial dos dados e pela presença de fortes correlações entre variáveis.

Esses fatores podem dificultar a formação de clusters bem definidos, sugerindo a necessidade de investigar outras técnicas de clusterização, como algoritmos baseados em densidade (por exemplo, DBSCAN) ou em modelos probabilísticos (como Gaussian Mixture Models). Além disso, seria interessante testar métricas de validação complementares e ajustar parâmetros de modelagem para garantir maior robustez nos resultados.

Pontos de Melhoria:

- Explorar métodos não lineares de agrupamento, como **Gaussian Mixture Models** (GMM), para capturar distribuições mais complexas.
- Utilizar técnicas de seleção de features para reduzir as redundâncias entre as variáveis.
- Testar algoritmos de clustering hierárquico completo para comparar a robustez dos agrupamentos.

CONCLUSÃO E TRABALHOS FUTUROS

Síntese

Este trabalho buscou identificar grupos similares em um conjunto de dados utilizando métodos de clusterização hierárquica e K-Means. A análise revelou que os dados poderiam ser representados adequadamente por 3 clusters, com mais de 50% da variância capturada pelos três primeiros componentes principais o que demonstrou que o agrupamento não supervisionado com K-means é uma abordagem viável para reconhecimento de atividades humanas, mesmo com dados de alta dimensionalidade. A análise exploratória e a seleção cuidadosa de hiperparâmetros foram cruciais para o sucesso da análise.

Trabalhos Futuros

- Implementar modelos mais avançados, como GMM (Gaussian Mixture Models) para comparar a qualidade dos clusters.
- Explorar técnicas de seleção automática de variáveis para reduzir a dimensionalidade sem perda de informação.
- Aplicar modelos supervisionados para comparar a acurácia em relação ao agrupamento não supervisionado.

REFERÊNCIAS

1. Dataset *Human Activity Recognition Using Smartphones*: UCI Machine Learning Repository.
2. Scikit-Learn Documentation: https://scikit-learn.org.
3. J. Pedregosa et al., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, 2011.