PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
NÚCLEO DE EDUCAÇÃO A DISTÂNCIA
Pós-graduação *Lato Sensu* em Ciência de Dados e Big Data

Daniela Matta Machado

ANALISE DOS DADOS DE VENDA – AMAZON USA
DEPARTAMENTO DE MARKETING

Belo Horizonte
Maio de 2022

Daniela Matta Machado

ANALISE DOS DADOS DE VENDA – AMAZON USA
DEPARTAMENTO DE MARKETING

Trabalho de Conclusão de Curso apresentado ao Curso de Especialização em Ciência de Dados e Big Data como requisito parcial à obtenção do título de especialista.

Belo Horizonte
Maio de 2022

SUMÁRIO

1. Introdução

1.1. Contextualização

O presente projeto objetiva o conhecimento, analise e preparo de uma massa de dados da Amazon-USA visando a transformação dessa grande quantidade de dados brutos em informações preciosas para os Gestores e/outros Tomadores de Decisões. Os dados avaliado foram gerados ao longo de 2019 e parte 2020 e as planilhas foram disponibilizadas no site da Kaggle.

Neste projeto os algoritmos escolhidos seguem a vertente ‘Aprendizagem de maquinas sem supervisão` pois não foi disponibilizado os dados de saida ou a variável de resposta. Assim, foi trabalhado os algoritmos Kmeans, DBSCAN e Kmeans-Shift para se encontrar o melhor número de agrupamentos dos clientes segundo os valores gastos nas compras, frequencia de compras e data da ultima aquisição.

1.2. O problema proposto

O principal objetivo do trabalho foi segmentar os clientes para gerar insights para o Departamento de Marketing da Amazon direcionar campanhas mais assertivas.

Os dados 2019 da Amazon – USA foram trabalhados segundo ‘Aprendizagem de máquinas não supervisionado’ por não apresentaram a variável resposta.

Algumas perguntas de Negocio tais como ‘Ano e mês de melhor faturamento’ , ‘Cidade com maior faturamento’ e ‘Produtos mais vendidos’ foram respondidos e demais planilhas foram tratadas para que a segmentação de clientes pudesse ser feita apartir dos dados de faturamento, frequência das compras e Data das últimas aquisições.

2. Coleta de Dados

Os dados para o trabalho foram coletados no Site da Kaggle neste link tendo sido importados para o Jupyter Notebook e trabalhados na linguagem Python.

As planilhas contendo informações como ID dos pedidos (Order ID), produto (Product), quantidade comprada (Quantity Ordered), preco (Price Each), data da compra (Order date) e o endereço da entrega (Purchase Address) foram importados

como tipo Objetc o que necessitaram de tratamentos ao longo do projeto. Ver tabela abaixo:

Nome da coluna/campo	Descrição	Tipo
Order ID	ID exclusivo de dada compra realizada no Amazon.	Object
Product	Produto vendido	Object
Quantity Ordered	Quantidade do item adquirido para cada Order ID	Object
Price Each	Preço Unitário de cada item	Object
Order Date	Data da compra.	Object
Purchase Address	Endereço do comprador	Object

3. Processamento/Tratamento de Dados

As 12 planilhas no formato CSV (1 planilha para cada mês do ano) foram importadas e consolidadas em apenas 1 planilha (VendasConsolidado). Apos a consolidação, iniciou-se o processo de conhecimento dos dados através do método info() do Pandas.

Figura01 : Tipo de dados inportados e quantidades

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 186850 entries, 0 to 25116
Data columns (total 6 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   Order ID          186305 non-null  object
 1   Product           186305 non-null  object
 2   Quantity Ordered  186305 non-null  object
 3   Price Each        186305 non-null  object
 4   Order Date        186305 non-null  object
 5   Purchase Address  186305 non-null  object
dtypes: object(6)
memory usage: 10.0+ MB
```

As 6 colunas do DataSet consolidado possuem 186.305 dados do tipo Object e mais de 10 MB de memória.

Para ‘Dados Missing’ a função isnull() foi utilizada juntamente com a função sum() para retornar o total de dados faltantes em cada coluna (545 dados nulos para cada coluna). Optou-se então pela exclusão dos mesmos uma vez este número não chegar nem mesmo a 1% do total de dados. Para exclusão de todas as linhas nulas foi usado a função dropna.

Para descobrir os ‘Valores únicos’ de cada coluna aplicou-se um ‘for’ para cada coluna com a função nunique().

Figura02 : Valores unicos dos dados

```
#Descobrir os valores unicos
ValoresUnicos = VendasConsolidado.select_dtypes(['category', 'object']).columns
for uniq in ValoresUnicos:
    print('{} : {} valores unicos'.format(uniq, VendasConsolidado[uniq].nunique()))
```

```
Order ID : 178438 valores unicos
Product : 20 valores unicos
Quantity Ordered : 10 valores unicos
Price Each : 24 valores unicos
Order Date : 142396 valores unicos
Purchase Address : 140788 valores unicos
```

A informação de 140.788 endereços diferentes foi considerada importante no trabalho uma vez que ela representara a identidade unica dos clientes (ID clientes) .

A função do Pandas isna() foi utilizada para checar valores ausentes (NaN) e as funções astype(str), astype(int) e astype(float) foram usadas para converter as colunas ‘Quantity Ordered’ e ‘Price Each’ do tipo Object para String e Float respectivamente. Em seguida, essas colunas foram multiplicadas para resultar na coluna ‘ValorVenda’.

Figura03 : Construcao da coluna ‘ValorVenda’ partir das colunas ‘Qde Produto’ e ‘preco unitario’

```
VendasConsolidado['ValorVenda'] = VendasConsolidado['Qtd produto'] * VendasConsolidado['Preco unitario']
VendasConsolidado['ValorVenda'].round(decimals=2)
VendasConsolidado.head()
```

Na coluna 'Order date' foi necessário separar as informações ano, mês e dia para melhor responder as perguntas do negócio. Para tal, aplicou-se as funções dt.year, dt.month e dt.day .

Figura04 : Tratamento das informações da coluna ‘Order Date’

```
#Separar a coluna 'Order date' em ano, mes e dia para melhor responder as perguntas do negocio

VendasConsolidado['Order Date'] = pd.to_datetime(VendasConsolidado['Order Date'])
VendasConsolidado['Ano'] =  VendasConsolidado['Order Date'].dt.year
VendasConsolidado['Mes'] = VendasConsolidado['Order Date'].dt.month
VendasConsolidado['Dia'] = VendasConsolidado['Order Date'].dt.day

VendasConsolidado.head()
```

O mesmo tratamento em separar as informações em colunas separadas foi necessário para ‘Purchase Address’ aplicando o lambda e split resultando assim nas colunas 'Cidade’, ‘Rua’ e ‘Estado’.

Figura05 : Tratamento das informações da coluna ‘Cidade’,’Rua’ e ‘Estado’

```
VendasConsolidado['Cidade'] = VendasConsolidado['Purchase Address'].apply(lambda x: x.split(',')[1])
VendasConsolidado['Rua'] = VendasConsolidado['Purchase Address'].apply(lambda x: x.split(',')[0])
VendasConsolidado['Estado'] = VendasConsolidado['Purchase Address'].apply(lambda x: x.split()[-2])

VendasConsolidado.head()
```

4. Análise e Exploração dos Dados

Na sequência do trabalho, após o tratamento nos dados, algumas perguntas do negócio já puderam ser respondidas tais como ‘Ano/Mês com maior vendagem’ , ‘Cidade que mais comprou’ entre outras.

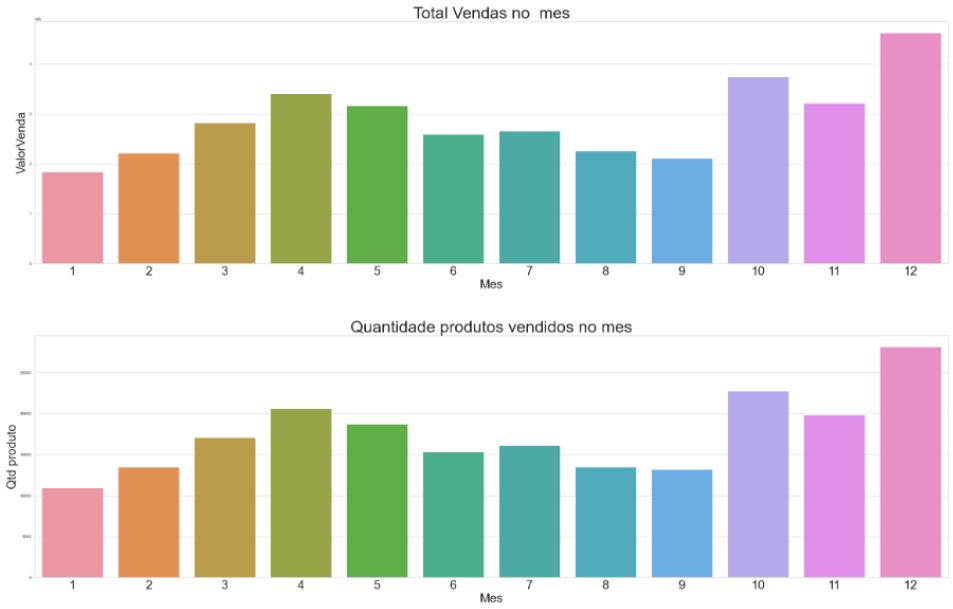
A função groupby juntamente com a função sum() na coluna ‘Ano’ (.groupby(['Ano']).sum()) resultou que o ano de 2019 ,comparado com 2020 (este ano apresentando apenas os primeiros mêses iniciais) teve maior faturamento (mais de 34 milhões de dolares).

Ano	ValorVenda
2019	34483365
2020	8670

Para responder qual foi o ‘Melhor mês no faturamento’ e ‘melhor mês de vendas de itens’ a mesma função groupby() e sum() foram usadas para a coluna ‘Mês’ . Foi visto que os 3 melhores mêses em termos de faturamento foram aqueles próximos do natal sendo que apenas dezembro faturou-se 4.613.443 milhões.

Dezembro tambem atingiu a marca de maior quantidade de itens vendidos. Para melhor visualização optou-se por construir os graficos barplot do seaborn:

Figura06 :Grafico de barras para Mes X Faturamento e Mes x Quantidades itens vendido



Para responder a pergunta de “Qual produto mais vendido’ um groupby foi usado na coluna ‘Product’ para agrupar e depois somar as quantidades. O top5 produtos mais vendidos (nlargest(5)) mostrou que pilhas AAA foram as campeãs com 31.017 pacotes vendidos seguidas por pilhas AA (27.635 pacotes vendidos).

```
Product
AAA Batteries (4-pack)      31017
AA Batteries (4-pack)       27635
USB-C Charging Cable        23975
Lightning Charging Cable    23217
Wired Headphones            20557
Name: Qtd produto, dtype: int32
```

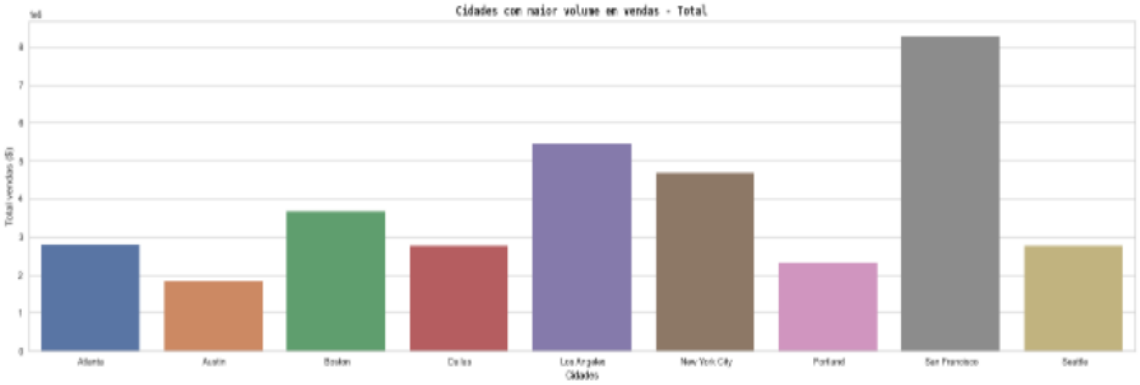
Para os preços unitários dos produtos foi usado a função groupby para agrupar os produtos, mean() para tirar a média dos valores obtidos e nlargest(5) para apresentar as top5. O Macbook Pro Laptop tem o maior valor médio unitario (1.700.00 dólares).

```
Product
Macbook Pro Laptop    1700.00
ThinkPad Laptop        999.99
iPhone                 700.00
Google Phone           600.00
LG Dryer               600.00
Name: Preco unitario, dtype: float64
```

E por fim, para ter a resposta de ‘Cidade de com maior faturamento’ tambem foi construido e analisado um gráfico de barras Barplot () do pacote seaborn (sns).

As cidades de San Francisco, Los Angeles e New York foram as que apresentaram maiores valores em Vendas .

Figura07 :Grafico de barras para Cidade X Faturamento



5. Criação de Modelos de Machine Learning

O presente trabalho utilizou os algoritmos de aprendizado não supervisionado para clusterização (agrupamento) . Os algoritmos que fazem parte do grupo ‘Não supervisionado” possibilitam, apartir dos dados de entrada sem conhecimento dos dados de saida, agrupamentos dos dados (n-elementos) por padrões e similaridades.

Para o trabalho foi utilizado a categoria de agrupamento de clustering particional. Não foi usado cluster hierarquico. No modelo particional, os clusters são gerados utilizando um critério pré definido do negócio não existindo relação hierarquica entre eles.

Neste projeto o primeiro algoritmo usado foi o k-Means. O kmeans simplifica o processo com uma ‘adivinhação’ inicial para essa distribuição (determina um k inicial) e em seguida modifica para checar se as alterações melhoram a homogeneidade dos elementos dentro do grupos. Ou seja, no inicio do treinamento do algoritmo ele não sabe quais são os dados dentro de cada cluster. Assim ele começa com o processo aleatório de determinar os pontos centróides dos grupos e depois ele vai calculando as distâncias entre os centros e os pontos. Caso seja necessário, ele vai mudando os centróides de forma que os dados ate os mesmos tenham a distância minimizada.

Este trabalho não teve restrições de negócio quanto ao número de clusters. Portanto, foram utilizados os dados de endereço como ID do consumidor, a frequencia de realização das compras, o valor total gasto por endereço e as recorrencias nas compras.

Para os valor inicial de K foi utilizado o metodo de Elbow para determinar a quantidade ideal de clusters e a metrica de Silhoete.

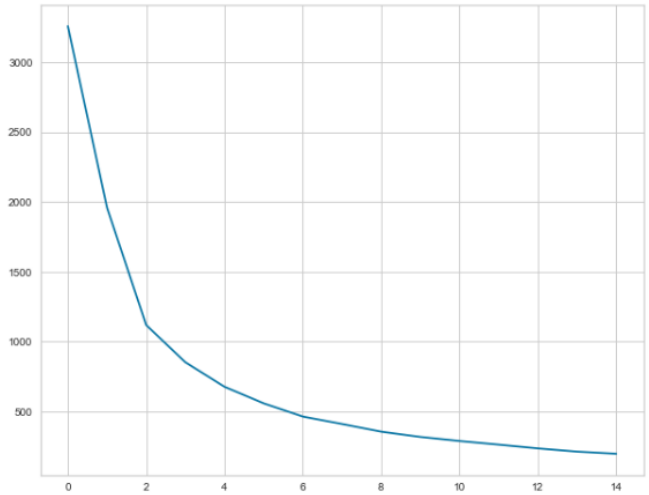
K-Means:

Determinou-se um range de clusters (de 1 a 15), criou-se o modelo com Kmeans e fez-se o treinamento (fit) do modelo de Kmeans com a amostra inicial (amostra3). Os Kmeans calculados para o range retornam uma lista ssd onde foi aplicado o método inertia para cálculo dos valores de clustres segundo Elbow.

Figura08: Modelo Kmeans para range de Ks e plot da lista gerada aplicando-se Elbow (inertia)

```
# k-means utilizando um numero aleatorio de clusters k
#Estou criando o modelo com Kmeans e aplicando o modelo com o fit no meu conjunto amostra3
# range de K
# Elbow-curve/SSD

lista = []
k_r = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]

for X in k_r:
    modelo = KMeans(n_clusters=X, max_iter=50)
    modelo.fit(amostra3)
    lista.append(modelo.inertia_)

# plot da lista
plt.plot(lista)
```

Figura09 : Plot – Metodo de Elbow



Observando a curva de Elbow acima vemos uma diminuição na distancia entre os dados e os centróides (eixo y) com a aumento dos clusters (eixo x). Dessa forma apartir do k=8 o valor torna bastante interessante.

O metodo de Silhouette tambem foi realizado para fazer um contra-ponto com o metodo de Elbow na determinacao do melhor valor para K.

O valor de silhouette [1, -1] sendo o valor mais proximo de 1 indicando que o dado tem muita similaridade com seu grupo e baixa similaridade com o grupo vizinho.

Para o range de K determinado no projeto o silhouette score foi mais positivo para k=12.

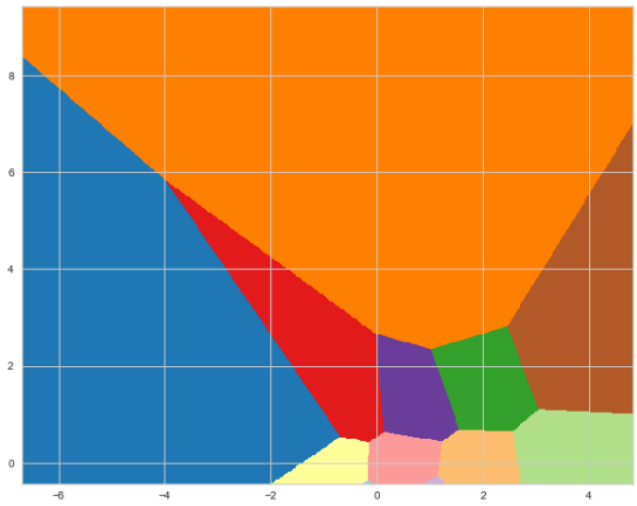
Figura10 : Analise de Silhouette para os vaores de 8 a 15 no range

```
# Analise de Silhouette -usar mesmo range de K do Elbow


k_r = [8,9,10,11,12,13,14,15]

for X in k_r:

    modelo = KMeans(n_clusters=X, max_iter=50)
    modelo.fit(amostra3)
    labels = modelo.labels_
    silhouette = silhouette_score(amostra3, labels)
    print("para X={0}, o score de silhoette {1}".format(X,  silhouette))
```

```
para X=8, o score de silhoette 0.4475496932566367
para X=9, o score de silhoette 0.4534754491132862
para X=10, o score de silhoette 0.4581780955853613
para X=11, o score de silhoette 0.42798021417616633
para X=12, o score de silhoette 0.46978935937380384
para X=13, o score de silhoette 0.44186122613555784
para X=14, o score de silhoette 0.4573370321144012
para X=15, o score de silhoette 0.4588265163226249
```

Assim trabalhou-se com k=12 e plotou-se o grafico meshgrid obtendo a seguinte figura:

Figura11 : Grafico mostrando 12 clusters



Observa-se na figura acima 12 regiões bem marcadas (12 clusters) onde se encontram os dados trabalhados no projeto.

Finalmente para K =12 foi construido o cluster-map. Criamos uma lista com as vari áveis trabalhadas ['Purchase Address','ValorVenda', 'Frequencia', 'Diff'] e criamos o cluster map para visualizar os dados que fazem parte dos respectivos clusters. Apartir daqui fazendo filtros pode-se responder várias perguntas de negócio como quantidade de cidades nos clusters que apresentam maior número de consumidores, calcular a média do valor de vendas por cluster entre outras.

Figura12 : DataFrame com as colunas alvo do projeto e o cluster

```
:  1 cluster_map
```

[57]:

	Purchase Address	ValorVenda	Frequencia	Diff	cluster
9078	157 9th St, San Francisco, CA 94016	11.95	1	325	1
68577	537 Lincoln St, San Francisco, CA 94016	750.00	2	15	6
127118	911 Hickory St, Los Angeles, CA 90001	2.99	1	288	1
75897	584 Walnut St, Boston, MA 02215	14.95	1	20	0
50861	423 River St, San Francisco, CA 94016	14.95	1	182	10
...	...	...	...	...	...
110268	803 Pine St, Seattle, WA 98101	379.99	1	178	10
119879	865 Johnson St, Seattle, WA 98101	11.95	1	152	10
103694	762 1st St, Seattle, WA 98101	11.95	1	278	1
131932	942 Jackson St, Dallas, TX 75001	150.00	1	278	1
121958	879 Main St, Los Angeles, CA 90001	249.99	2	50	8

1407 rows × 5 columns

Realizando um groupby para as colunas cluster e ValorVenda aplicando a função mean() pode-se concluir que o cluster 7 tem o maior valor em faturamento seguido pelo cluster 11.

Figura13 : Calculo da media faturamento para os clusters

```
# Calcula a média ValorVenda por cluster
cluster_map.groupby('cluster')['ValorVenda'].mean()
```

```
cluster
0       45.438234
1       46.471269
2      618.992500
3     1270.472564
4      294.652574
5      460.748750
6      431.423210
7     1785.623750
8      112.888962
9      777.454667
10      58.270216
11    1726.961579
Name: ValorVenda, dtype: float64
```

Realizando um groupby para as clounas cluster e Purchase Address aplicando a função mean() pode-se concluir que o cluster 0 concentra o maior número de endereços .

Figura14 : Cálculo Números de endereço por clusters

```
#Agrupa os clusters por enderecos
cluster_enderecos = cluster_map.groupby('cluster')['Purchase Address'].count()
cluster_enderecos
```

```
cluster
0     334
1     260
2      28
3      39
4     136
5      96
6      81
7      16
8     106
9      60
10    232
11     19
```

Algoritmo DBSCAN:

O algoritmo DBSCAN não usa k-clusters previamente definidos como o Kmeas, mas ele tem os parametros proprios como ‘eps’ e o ‘min_samples’. O parâmetro eps é a distância máxima entre dois pontos de dados para serem considerados pontos de mesma vizinhança. O parâmetro min_samples é a quantidade mínima de pontos de dados no bairro para ser considerado um cluster.

Na utilização do DBScan foi feita a construção do modelo utilizando os parametros eps e min_samples padrões (eps = 0.2, min_samples = 5) gerando o primeiro modelo que, ajustando para a amostra (fit) apresentou -se da forma abaixo:

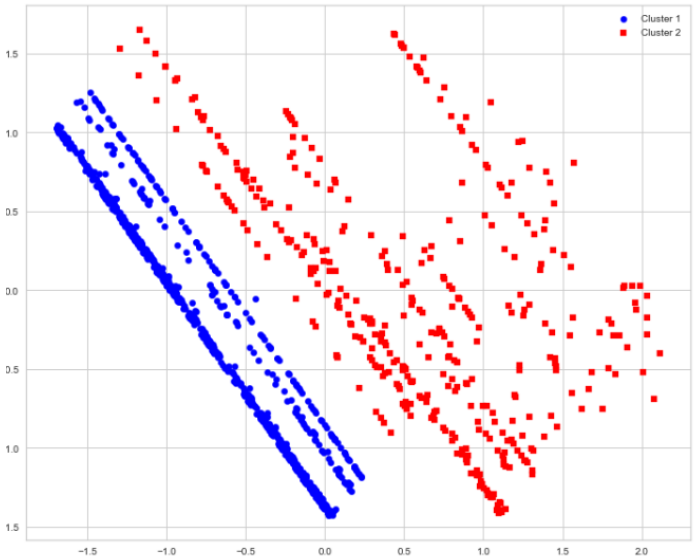
Figura15 : Construção do modelo utilizando DBSCAN

```
# Construção do modelo
# Estes parametros eps e min_samples sao padroes
# Primeira versão do modelo
modelo = DBSCAN(eps = 0.2, min_samples = 5, metric = 'euclidean')
```

```
# Fit do modelo
y_db = modelo.fit_predict(amostra3)
```

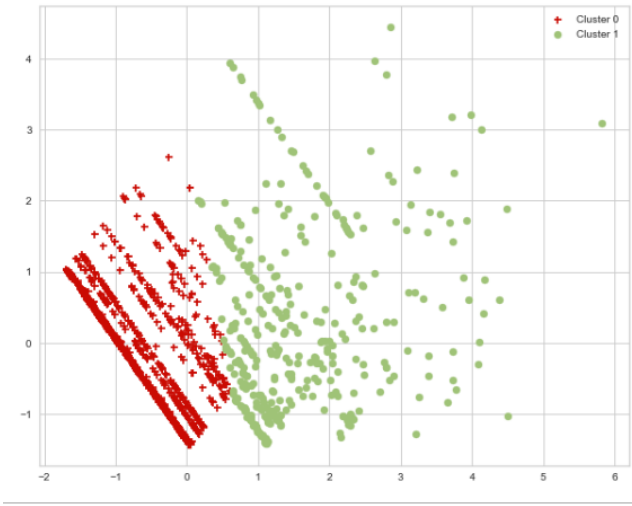
O DBSCan encontrou apenas 2 clusters que apresentaram a seguinte figura de plot:

Figura16 : Plot dos clusters - DBSCAN



Foi feito usando o Kmeans para k = 2 apenas para fins de comparação com os resultados obtidos no plot do DBScan.

Figura17 : Plot dos clusters utilizando Kmeans = 2



Algoritmo Kmean-Shift:

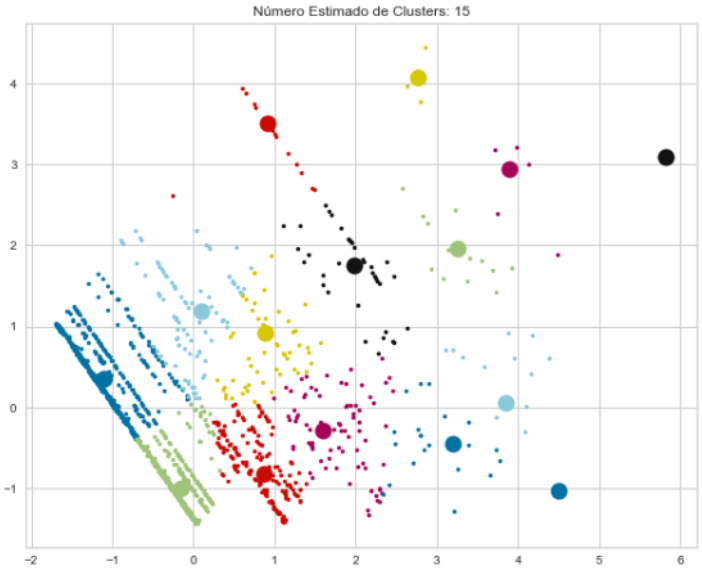
Por fim, o terceiro algoritmo utilizado foi o Kmean-Shift. Este algoritmo tambem utiliza o conceito de centróides locados nos pontos de maior densidade de dados. A vantagem do Kmean-Shift em comparação com o Kmean é não requer valores de clusters para iniciar mas requer parâmetros próprios como Bandwidth (descreve o tamanho da região), Seeds(usada para inicializar o kernel) e Bin_seeding (aceita valor booleano).

Figura18 : Construção do modelo utilizando Kmean-Shift

```
# Cria o modelo

# bandwidth = Comprimento da Interação entre os exemplos, também conhecido como a largura de banda do algoritmo.
bandwidth = estimate_bandwidth(amostra3, quantile = .1, n_samples = 500)

# Cria o modelo
modelo_v1 = MeanShift(bandwidth = bandwidth, bin_seeding = True)

# Treina o modelo
modelo_v1.fit(amostra3)
```

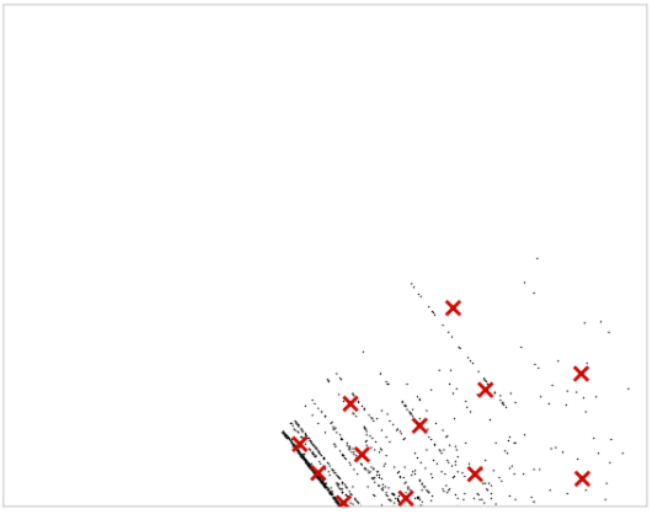
```
MeanShift
MeanShift(bandwidth=0.6535935644674339, bin_seeding=True)
```

Figura19 : Plot dos clusters utilizando algoritmo Kmean-Shift



Para fins de comparação foi feito para k = 15 o modelo com o Kmeans e o plot dos centroides encontrados mostrado abaixo:

Figura20 : Plot dos clusters utilizando algoritmo Kmean e k =15



Como conclusão do trabalho o algoritmo Kmeans, Kmean-shift e BDScan encontraram valores diferentes de clusters para a amostra. Os valores de Kmeans-Shift (15 clusters) e Kmeans (range entre 10 e 15) seriam os algoritmos mais interessantes a serem apresentados ao Tomadores de Decisão. Ainda que não tenha o ‘algoritmo ‘ mais certeiro, cabe ao Cientista de Dados fazer, juntamento com o Negocio, a melhor analise casando com os objetivos propostos. O número de 15 clusters de consumidores agrupados conforme os atributos de ‘Valor venda’ , ‘Frequencia de compras ‘ e ‘Numero de dias desde a ultima compra’ apresentou melhores valores para metrica Silhouete justificando assim a sua escolha.

6. Links

O projeto TCC – PUC – Cientista de Dados bem como as planilhas de dados Kagle e o video estão compartilhado no repositorio Git:

Link GitHub: https://github.com/Danielamms/TCC-CientistaDados-PUC

Link Youtube: https://youtu.be/VsptA2M974I