

# Análise dos datasets

Felipe B. Diniz

## 1 Introdução

Neste PDF encontram-se diversas análises dos datasets `sales.csv` e `comp_prices.csv` (que chamaremos apenas de *comp*), assim como a descrição dos métodos para a predição de quantidades a partir dos preços dados. Associado a este texto há o script para realizar as predições e um notebook com os códigos para as imagens que veremos aqui. Estes arquivos se encontram no meu github, em <https://github.com/felipebottega/Machine-Learning-Codes>, são os arquivos `analise.ipynb`, `analise.pdf`, `script.py`. Começamos visualizando algumas entradas iniciais dos dados.

```
1 sales.head()
```

	PROD_ID	DATE_ORDER	QTY_ORDER	REVENUE
0	P6	2015-08-02	1.0	1808.99
1	P6	2015-08-17	1.0	1674.00
2	P6	2015-08-17	1.0	1673.95
3	P6	2015-08-11	1.0	1674.00
4	P6	2015-08-17	1.0	1674.00

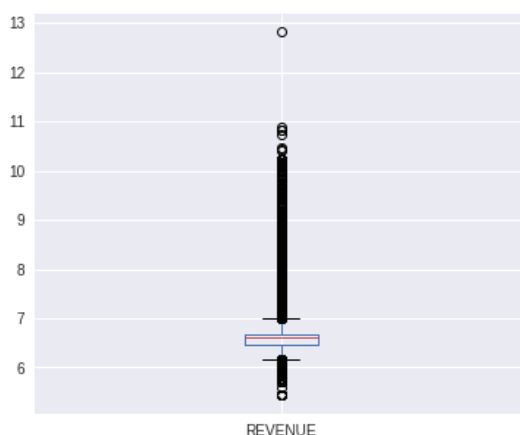
```
1 comp.head()
```

	PROD_ID	DATE_EXTRACTION	COMPETITOR	COMPETITOR_PRICE	PAY_TYPE
0	P6	2015-01-30 08:11:38	C3	1898.00	1
1	P6	2015-01-30 08:11:38	C1	1898.00	1
2	P6	2015-01-30 08:11:38	C6	1999.90	1
3	P6	2015-01-31 20:10:14	C2	1894.88	2
4	P6	2015-01-31 20:10:14	C3	1894.88	2

## 2 Limpeza de dados

### 2.1 Dados omissos e outliers

Primeiramente, podemos notar que não há dados faltando em ambos os datasets. Então estamos encerrados neste aspecto. Para se ter uma noção melhor dos outliers de algumas colunas escolhidas, usaremos a visualização por boxplot. Estamos fazendo o plot dos dados em escala logarítmica uma vez que há valores muito grandes entre os dados.



Baseado nestes gráficos, poderíamos remover os outliers segundo algum critério. Porém, optamos por manter os outliers, pois, na sua maioria, se tratam de vendas com valor alto. Não podemos negligenciar isto.

## 2.2 Ajuste das colunas

Consideramos desnecessário o horário da extração dos preços da competição. Além disso, algumas colunas se referem ao mesmo tipo de informação, então iremos renomeá-las como indicado abaixo:

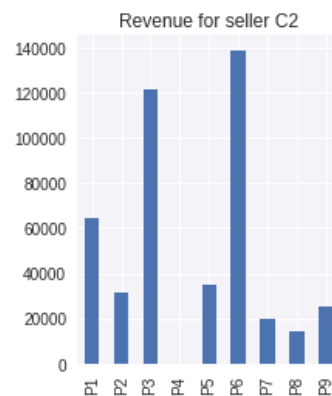
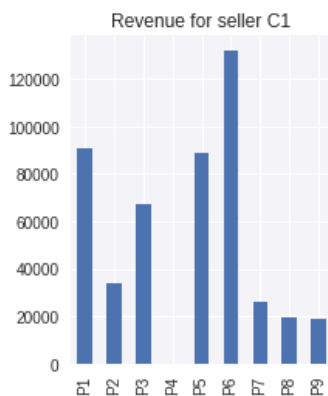
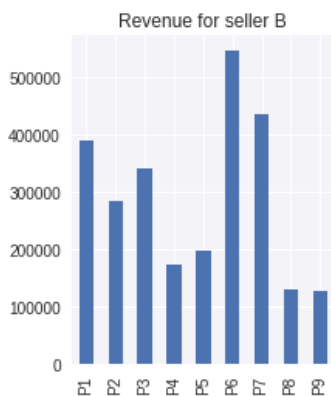
sales column name	comp column name	new name
DATE ORDER	DATE EXTRACTION	DATE
REVENUE	COMPETITOR PRICE	REVENUE
PROD ID	PROD ID	PROD

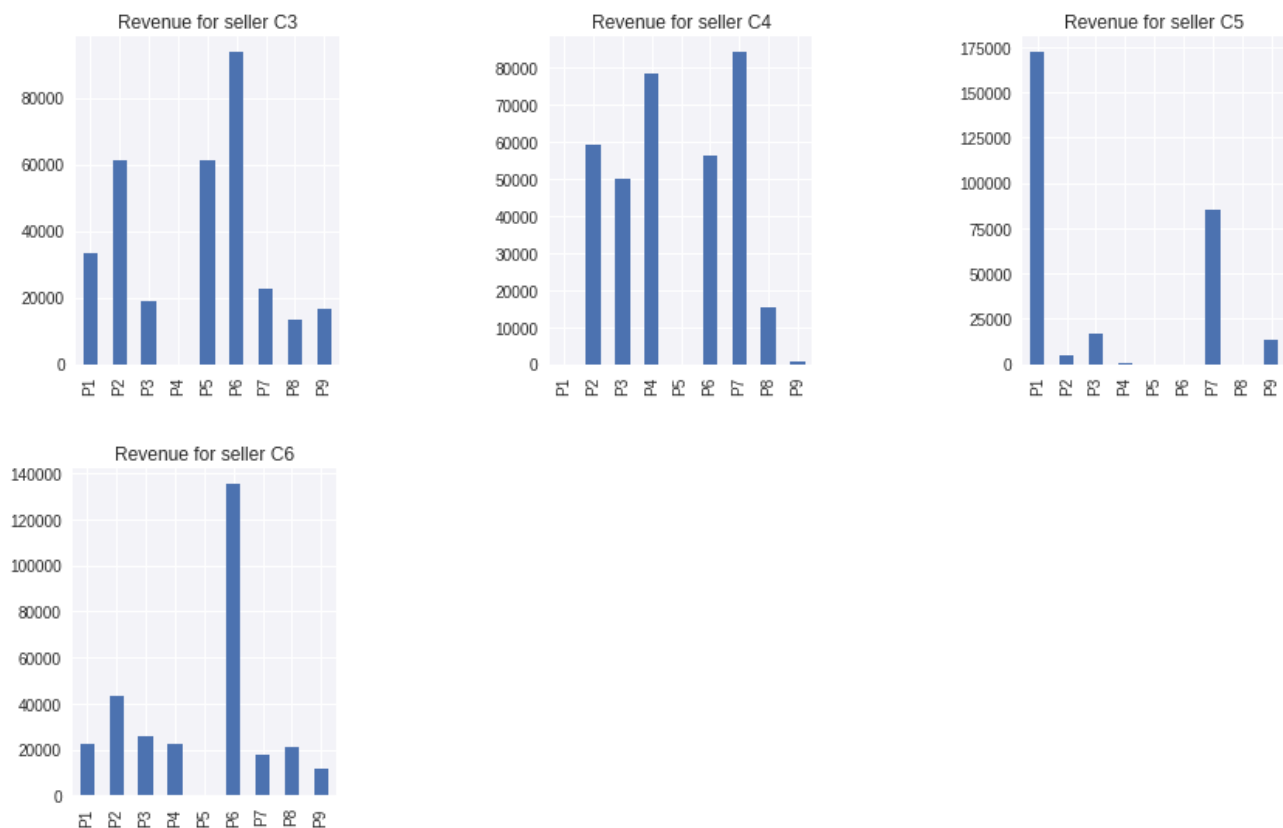
Logo no início do dataset *sales* podemos perceber que há um mesmo produto sendo vendido três vezes no mesmo dia, talvez até mais, mas vamos supor que seja três. O produto em questão é o P6, na data 2015-08-17. Como o nosso interesse é prever a quantidade de vendas por produto em cada momento, o correto seria que este produto aparecesse apenas uma vez, com o valor da coluna QTY ORDER sendo igual a 3. Antes de continuar devemos arrumar isso para todas as possíveis ocorrências.

## 3 Análise de venda dos produtos

Nesta parte, vamos ver quais produtos interessam mais a cada uma das empresas. O "interesse" aqui é medido pela coluna REVENUE, que nos diz o rendimento que a empresa obteve em cima do produto. Para isto, nos utilizamos da visualização de gráficos de barras.

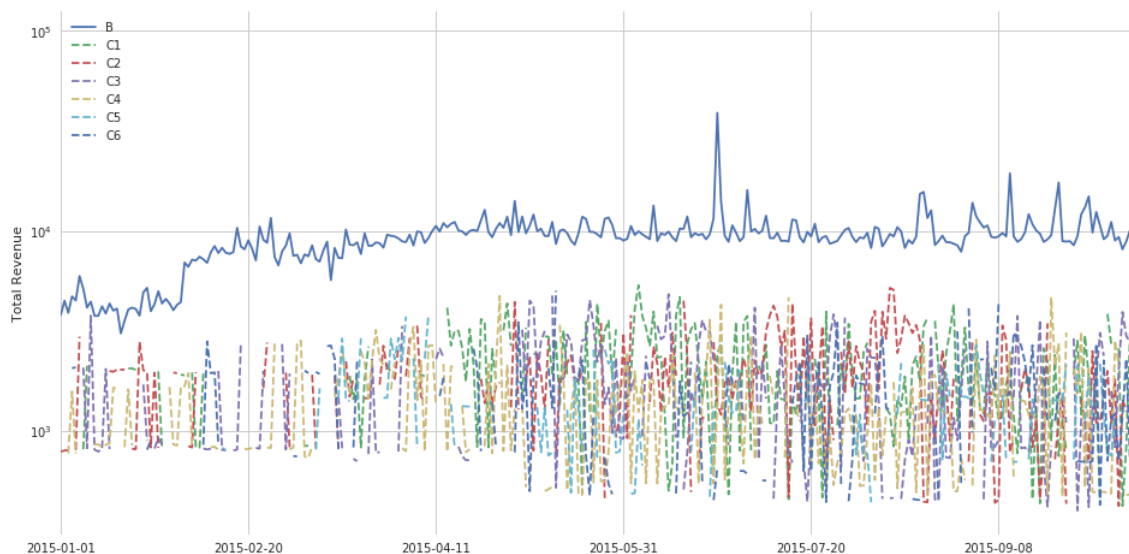
De maneira geral, os produtos de mais interesse são P6 e P7, com a exceção de P1 que é de bastante interesse para C5. A empresa B tem claro foco no produto P6. Pelo que podemos ver dos histogramas, todos os competidores, exceto C4 e C5, seguem a estratégia de vender mais o produto P6. O competidor C5 parece tentar vender mais P7 e P1, enquanto que o competidor C4 fica ficado em P7. Enquanto alguns tentam equilibrar o balanço de vendas, outros ficam totalmente focados em P6.



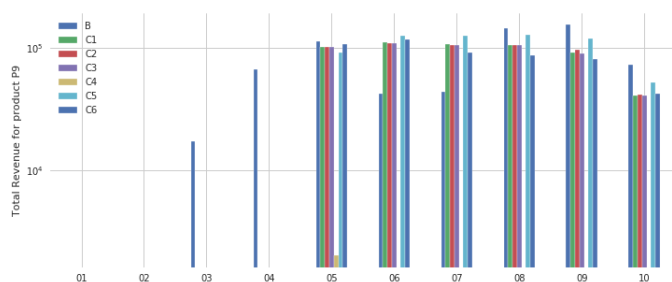
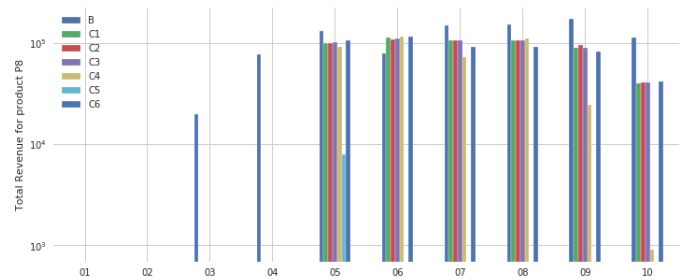
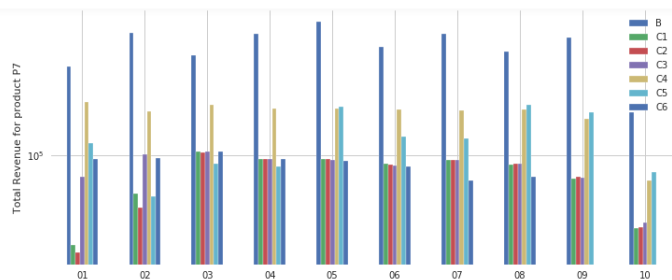
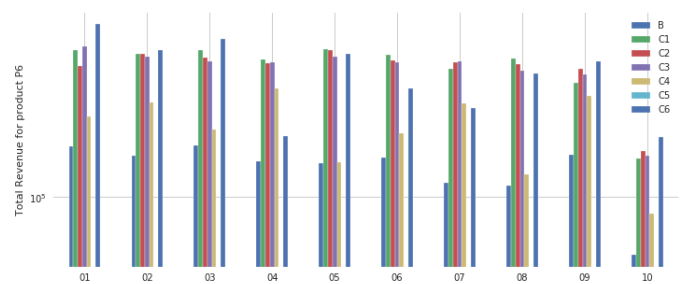
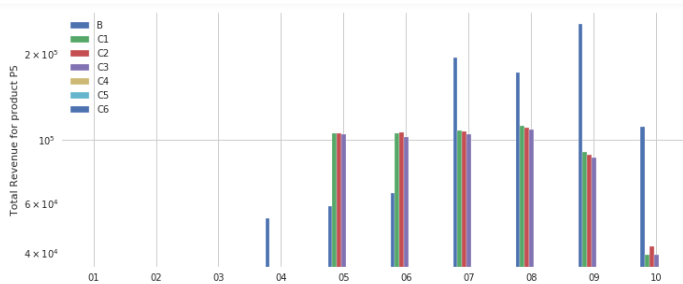
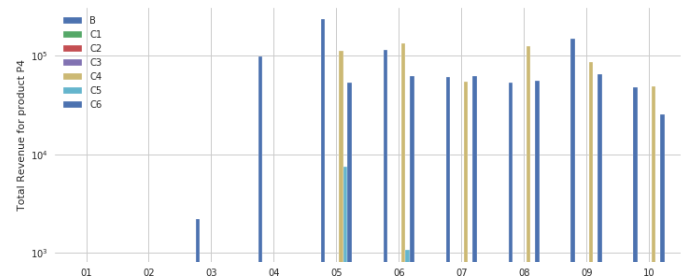
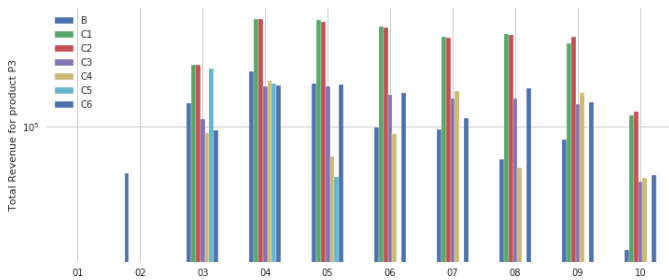
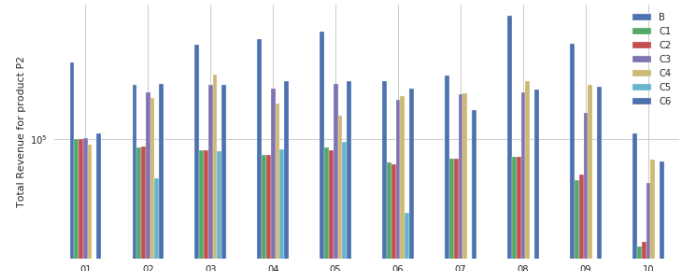
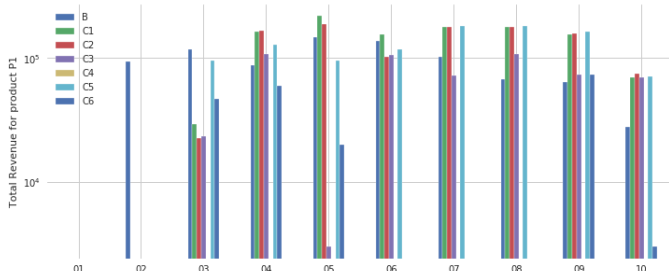


### 3.1 Séries temporais

Vimos os produtos que interessam mais a cada empresa, e a partir disso traçamos uma ideia geral da estratégia escolhida por cada empresa. Pelos valores mostrados acima, é possível notar que a empresa B lucra muito mais que as outras empresas. Isto pode não ser necessariamente um quadro exato da realidade pois os dados extraídos da competição representam apenas uma parcela de duas vendas. Outro dado interessante para ser extraído é o rendimento total da empresa em cada dia (a soma das vendas de todos os produtos em um dado dia).



Como há muita flutuação nos valores, talvez seja mais interessante agrupá-los mensalmente. Dessa maneira esperamos poder ver mais facilmente os padrões. Com uma quantidade menor de datas, é possível fazer gráfico de barras, o que é mais claro neste caso.



## 4 Clusterização com K-means

Já temos uma ideia geral de como é o dataset e como as features se relacionam entre si. Agora devemos preparar o dataset para o treinamento. Iremos proceder da seguinte maneira:

- As colunas serão PROD, DATE, QTY, REVENUE B, REVENUE C, PAY TYPE (consideramos toda a competição como uma coisa só) e MONTH
- Podemos ter no máximo 9 linhas com a mesma data, pois cada instância de uma data será a média dos rendimentos de um produto específico vendido naquele dia.
- O dataset será normalizado.

### 4.1 Junção dos datasets

Agora que temos tudo mais organizado, fazemos um join nas linhas em que há coincidência de dia e produto. Após isso, haverá perda de linhas do dataset *sales* quando tivermos algum dia e produto sem match no outro dataset. Neste caso, para evitar perder informação, usamos a informação da linha de *sales* como se fosse a linha da competição. Isto significa que nestes casos usaremos apenas a informação de sales para o aprendizado, mas ainda assim é melhor do que descartar a informação. No fim das contas, teremos 2162 inputs para treinar o programa. Esta é uma redução considerável, pois no início tínhamos cerca de 300000 inputs para lidar só em *sales*.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2162 entries, 0 to 2161
Data columns (total 7 columns):
PROD          2162 non-null object
DATE          2162 non-null object
QTY           2162 non-null int64
REVENUE_B     2162 non-null float64
REVENUE_C     2162 non-null float64
PAY_TYPE      2162 non-null float64
MONTH         2162 non-null int64
dtypes: float64(3), int64(2), object(2)
memory usage: 118.3+ KB
```

### 4.2 Normalização dos dados

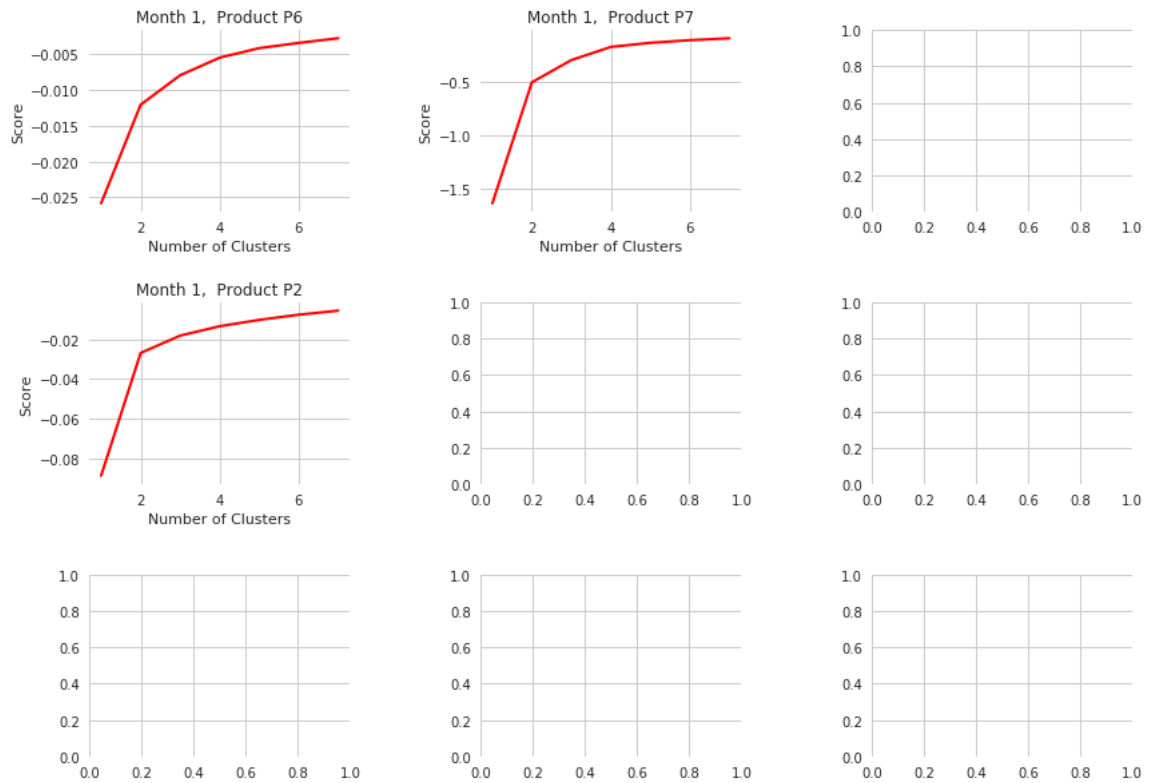
Além de converter todas as features em números, precisamos arrumar a escala destes números, pois a coluna REVENUE possui valores muito mais altos que todas as outras colunas. Se não arrumarmos isso, a clusterização será enviesada em favor dos valores das colunas (pois o dataset será praticamente unidimensional), com as outras features tendo papel pouco relevante no processo. Existem diversas maneiras de normalizar as features. Aqui nós usaremos a *normalização pela média*, dada pela fórmula abaixo.

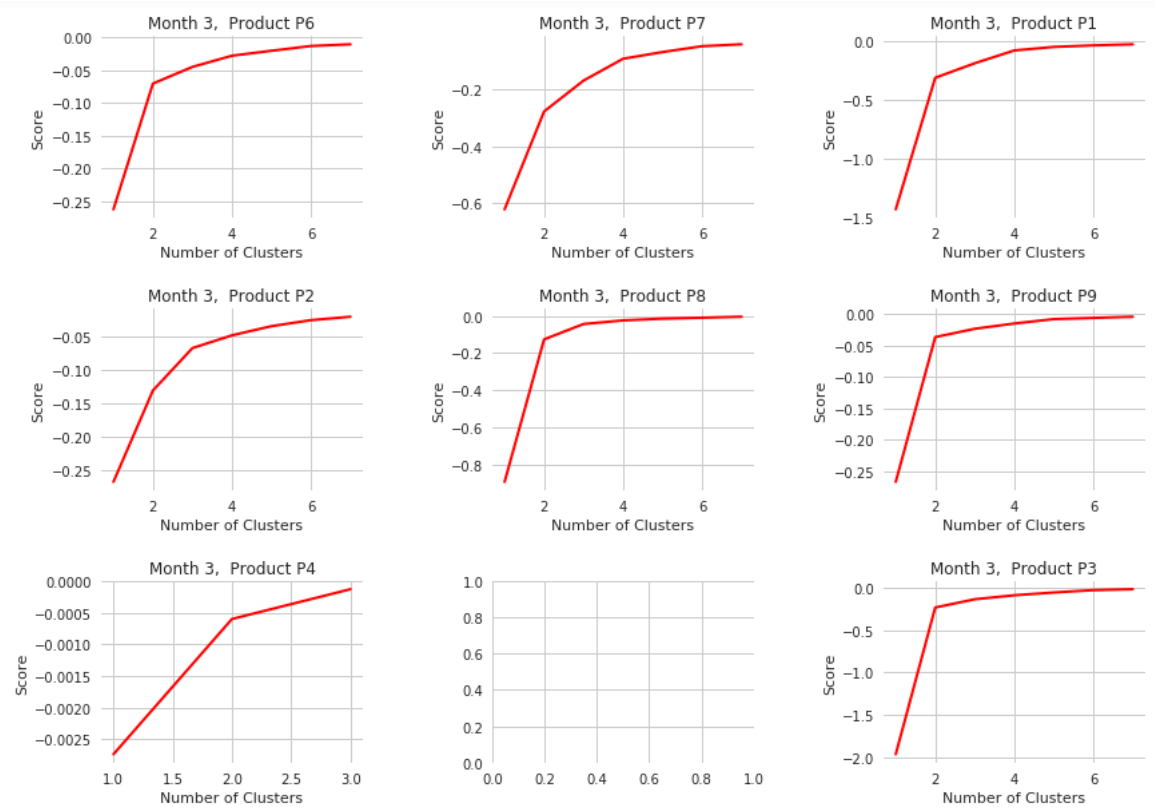
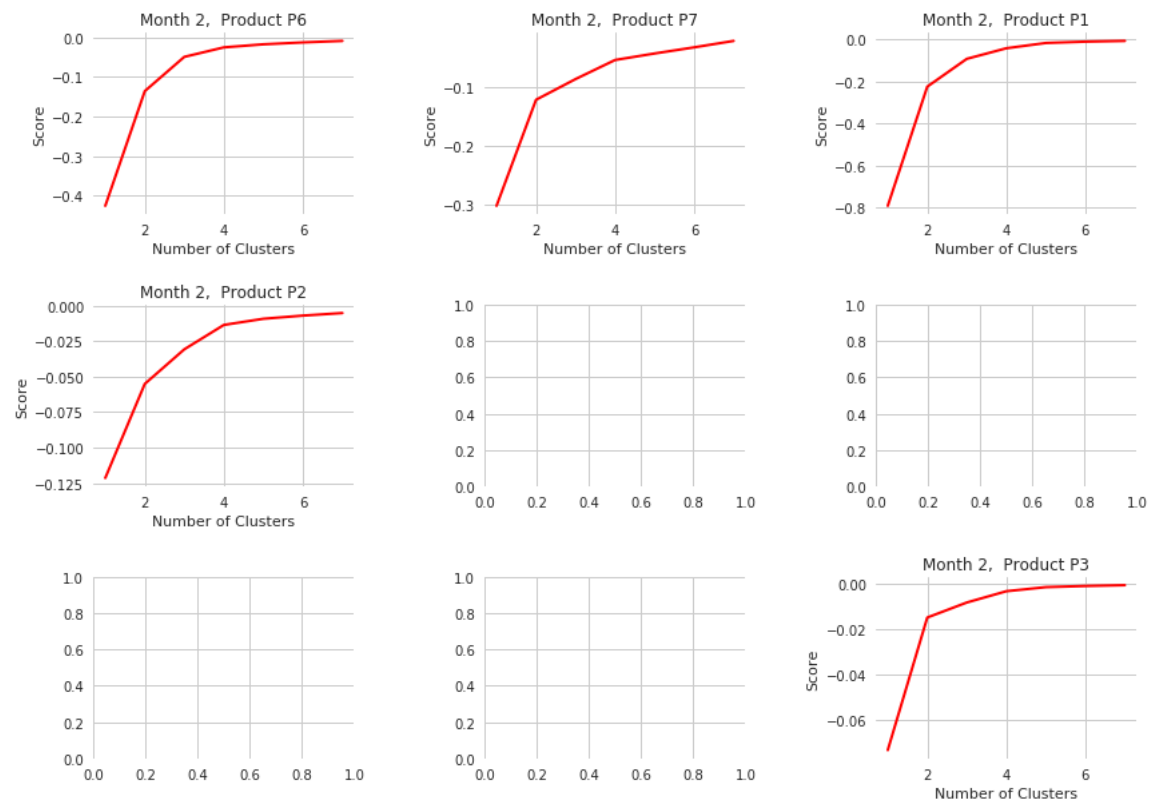
$$x_{new} = \frac{x - E[x]}{\max(x) - \min(x)}$$

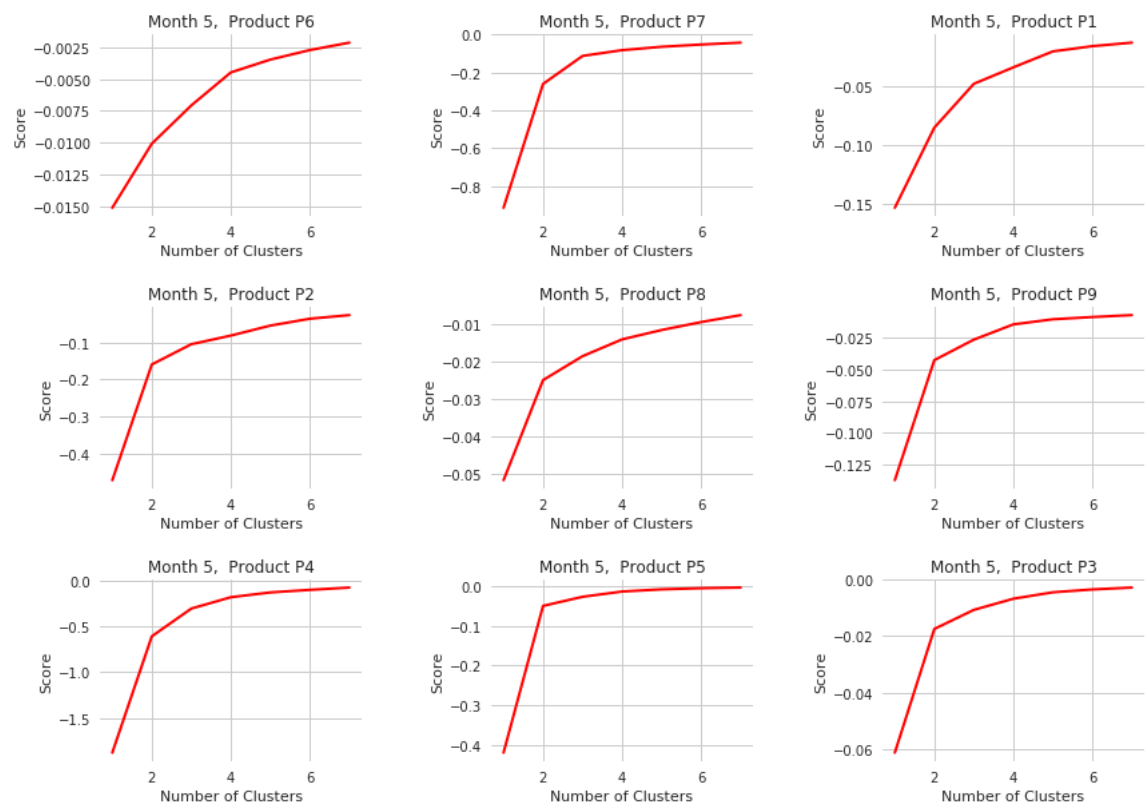
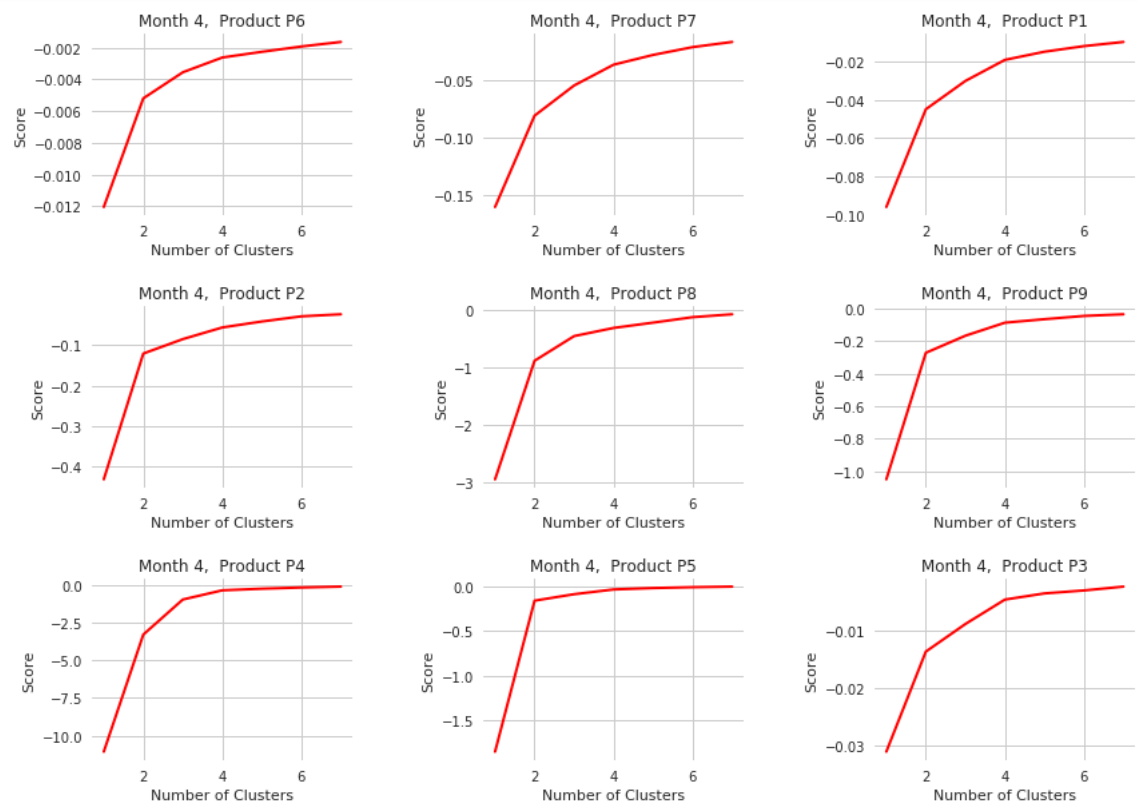
$E[x]$  é a média dos valores da coluna de  $x$ ,  $\max(x)$  é o máximo da coluna e  $\min(x)$  o mínimo da coluna. Faremos este procedimento para todas as colunas.

### 4.3 Treinamento

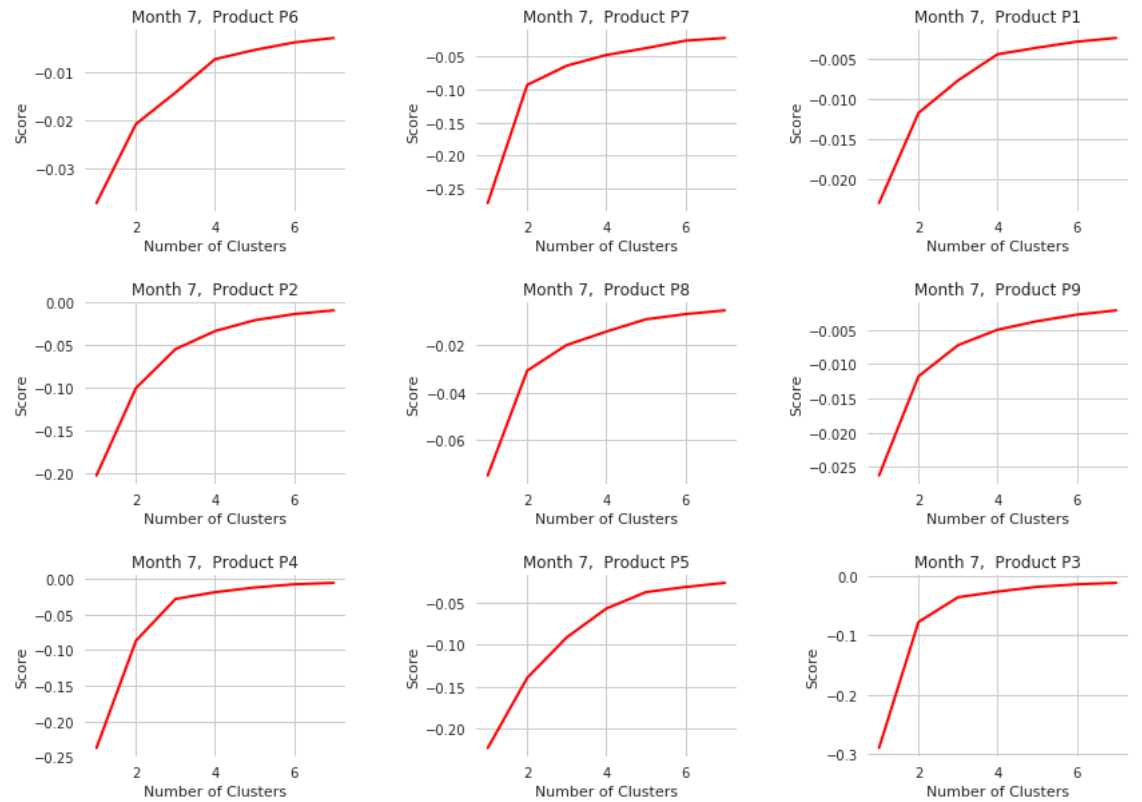
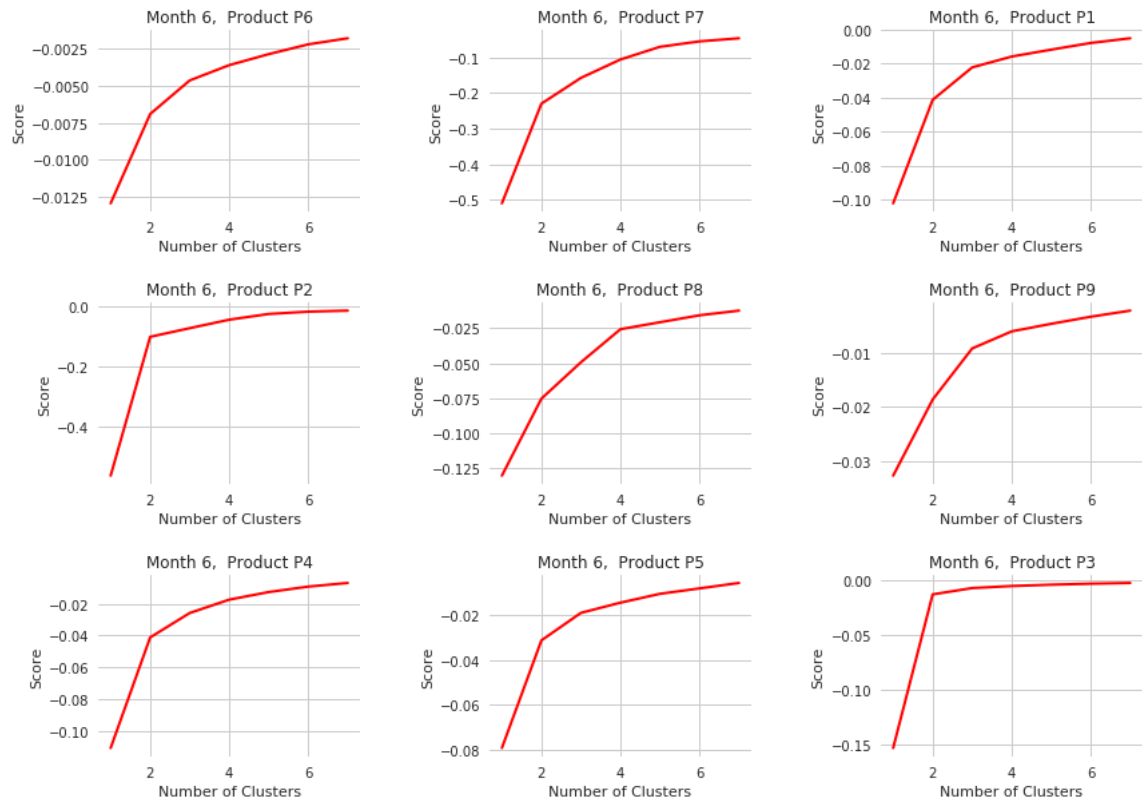
Para cada par (mês, produto) teremos um conjunto de clusters. Começamos agrupando os dados desta maneira. Para determinar o número de clusters, usamos o método *elbow* (mais detalhes em [https://en.wikipedia.org/wiki/Elbow\\_method\\_\(clustering\)](https://en.wikipedia.org/wiki/Elbow_method_(clustering))). Mostramos os gráficos dos resultados do treinamento abaixo.

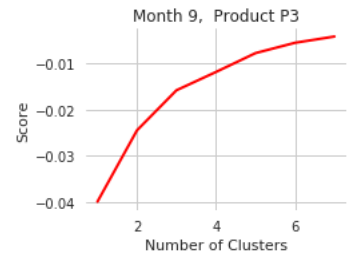
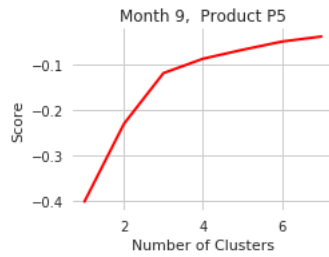
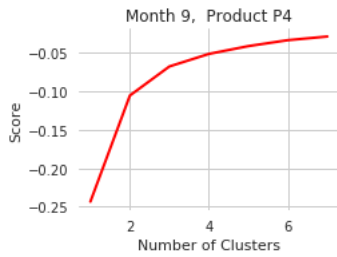
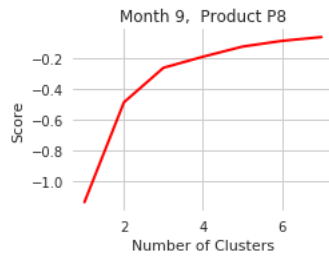
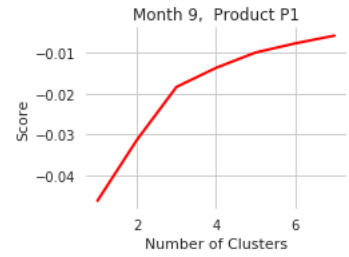
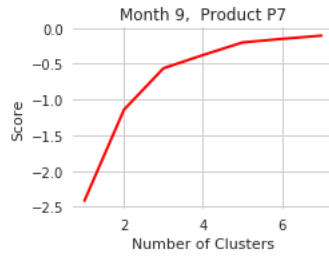
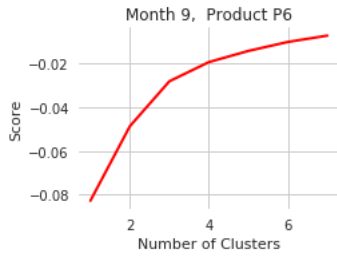
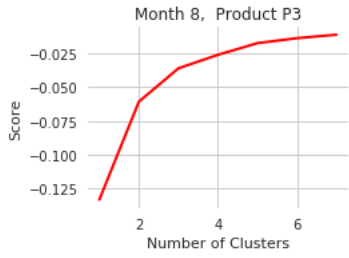
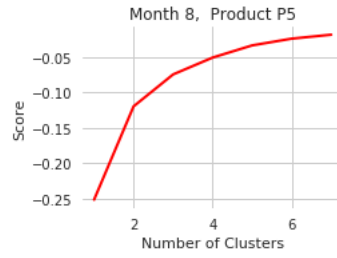
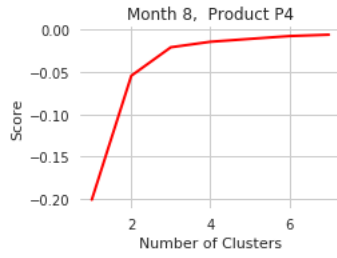
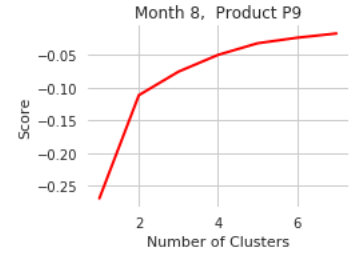
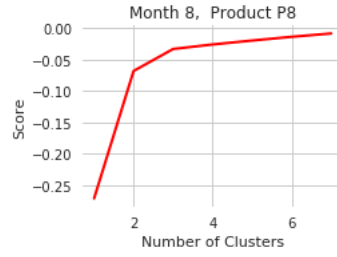
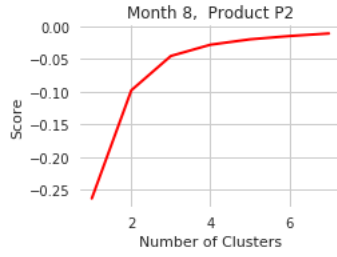
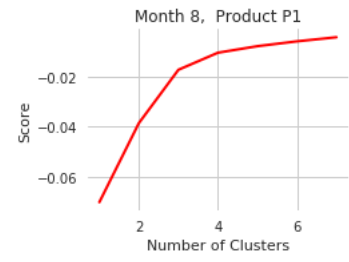
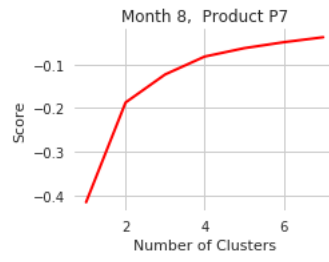


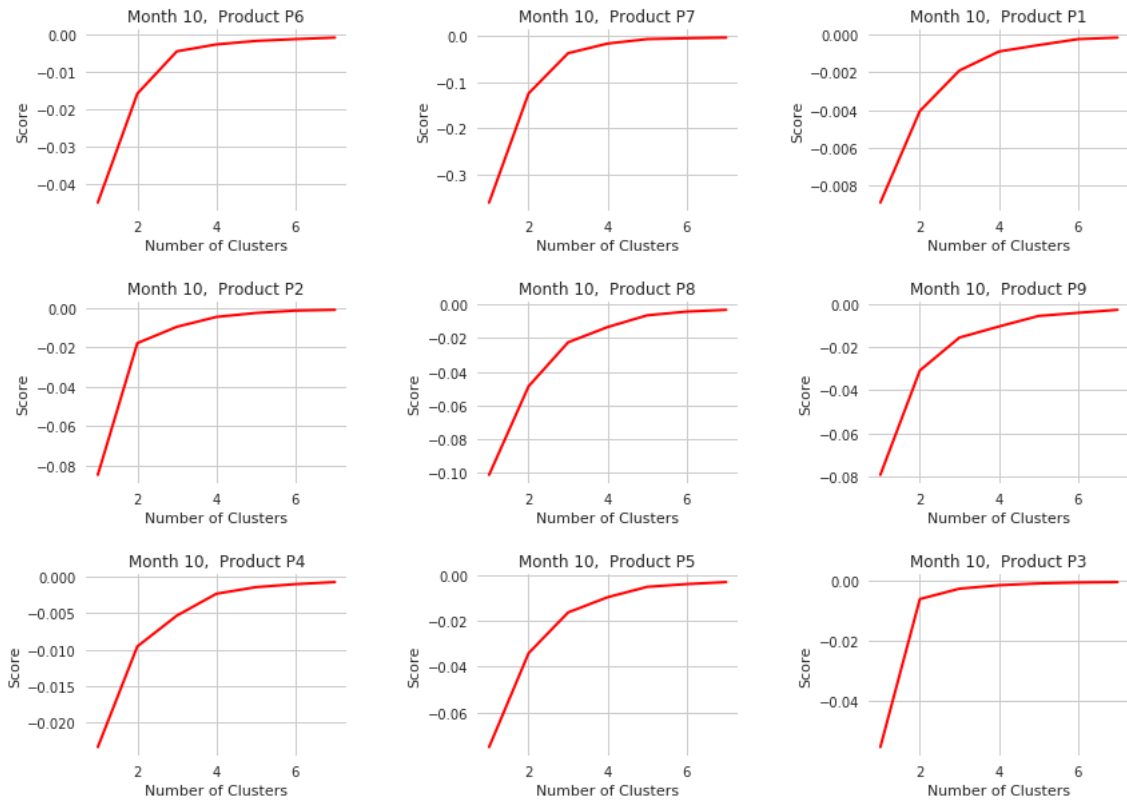








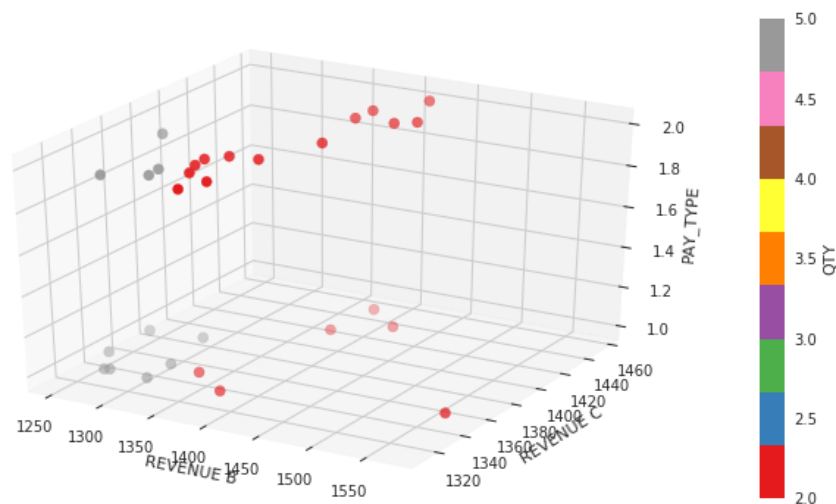




## 4.4 Predição

Dado o mês, o produto e os dois preços, verificamos qual o centróide mais perto das coordenadas dos preços. Este centróide está relacionado a uma certa quantidade que foi obtida durante o treinamento. Esta será a quantidade que usaremos para a predição.

Apenas para efeito de ilustração, vamos olhar o caso do mês 4 e produto P3. Lembre que cada amostra é um vetor tridimensional com as informações [QTY, REVENUE B, REVENUE C, PAY TYPE]. Iremos fazer um scatter plot no plano, onde a quantidade será monitorada através da cor dos pontos, obtidos durante o treinamento.



## 4.5 Acurácia do modelo

Agora que vimos que o algoritmo está separando bem os clusters, vamos como está a acurácia dele. Para isso, faremos previsões para todos os dados do dataset original, após o tratamento.

Sejam  $y^{(1)}, \dots, y^{(n)}$  aos targets originais do dataset, onde  $y^{(i)}$  é o elemento da linha  $i$  e coluna QTY, e sejam  $\tilde{y}^{(1)}, \dots, \tilde{y}^{(n)}$  as previsões dadas pelo nosso modelo. Uma maneira de se medir a acurácia do nosso modelo é usando a métrica RMLSE (Root Mean Squared Logarithmic Error). Esta métrica geralmente é usada quando não queremos penalizar grandes diferenças nos valores previstos e os reais quando os valores previstos e verdadeiros são grandes. Ela é definida por

$$\sqrt{\frac{1}{n} \sum_{i=1}^n (\log(1 + \tilde{y}^{(i)}) - \log(1 + y^{(i)}))^2}.$$

A grosso modo, esta métrica nos dá um fator de proporcionalidade entre o valor previsto e o real. Quanto maior for o valor, pior é a previsão, e quanto mais perto de zero, melhor é a previsão. Abaixo mostramos o plot da evolução do RMSLE. Podemos ver que a acurácia geral do modelo tende a melhorar conforme mais dados são inseridos nele.

