B2W CHALLENGE

Price Analysis

1. Problem

In the highly competitive Market, constantly the companies have to find ways to adjust or optimize the relationship between Price and Demand.

2. Goal

The main objective is to create a model to predict the quantity sold for each product given a prescribed price. Along with the statistical model, we need metrics, relationships and descriptions of these data in order to understand the sales behavior.

3. Methodology

To complete this challenge, I performed the following steps, according some steps to the CRISP-DM methodology:

- Data Understanding

- Data Preparation

- Modeling

- Evaluation

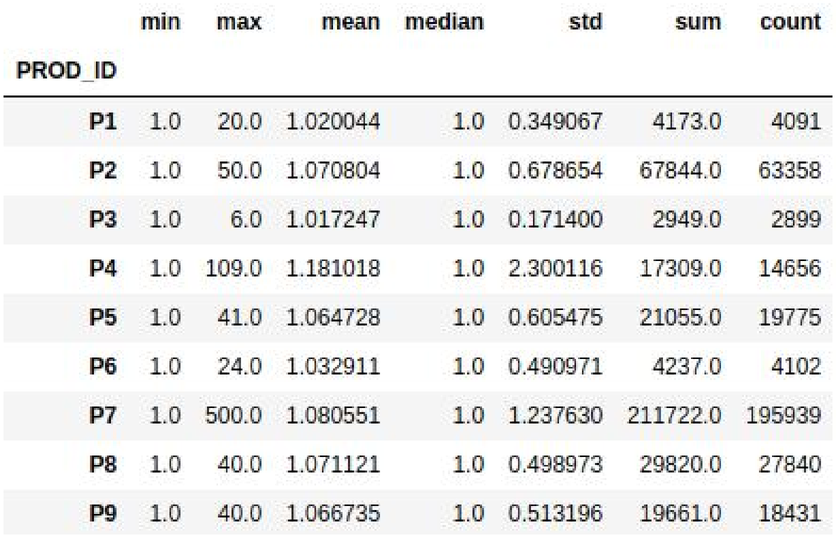
4. Data Understanding

The purpose of this step is to better understand the behavior of each product sold. For this, I applied several methods to find out more about imported data.

Following I will show some descriptive statistics of the analyzed products.

4.1. Analysis of Quantity Sold by Products

The following table shows some measurements of the quantity sold by product, as a minimum, maximum, mean, median, standard deviation and count.



PROD_ID	min	max	mean	median	std	sum	count
P1	1.0	20.0	1.020044	1.0	0.349067	4173.0	4091
P2	1.0	50.0	1.070804	1.0	0.678654	67844.0	63358
P3	1.0	6.0	1.017247	1.0	0.171400	2949.0	2899
P4	1.0	109.0	1.181018	1.0	2.300116	17309.0	14656
P5	1.0	41.0	1.064728	1.0	0.605475	21055.0	19775
P6	1.0	24.0	1.032911	1.0	0.490971	4237.0	4102
P7	1.0	500.0	1.080551	1.0	1.237630	211722.0	195939
P8	1.0	40.0	1.071121	1.0	0.498973	29820.0	27840
P9	1.0	40.0	1.066735	1.0	0.513196	19661.0	18431

Figure 1

By looking at the table above, you can see very large values in certain measurements as an example the product P7 that has the highest quantity of products sold in relation to the others. In second is the product P4 with maximum value of 109 quantities sold. Although its maximum value is much lower than the value of the product P7, the value of its standard deviation is much higher. This means that the variation in the quantity sold is further from the mean value.

For a better understanding, I will show below the data plotted in the chart.

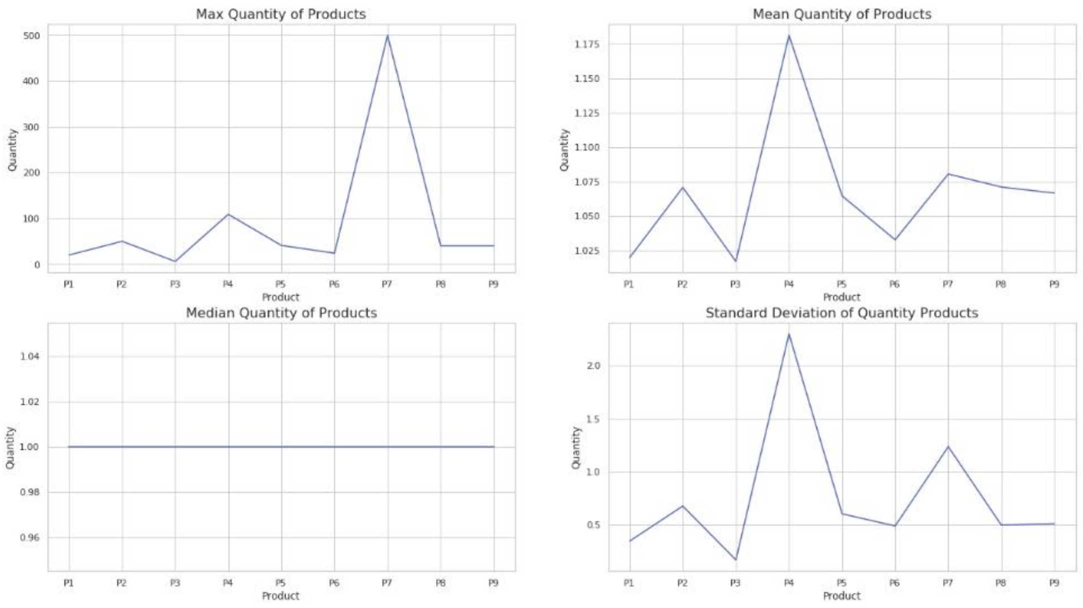
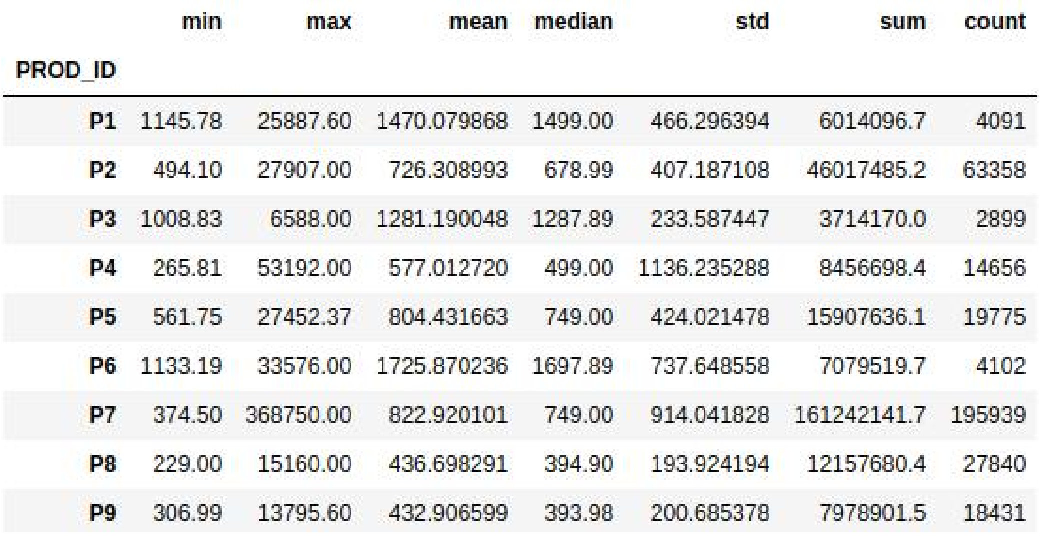


Figure 2

4.2. Analysis of Revenue by Products

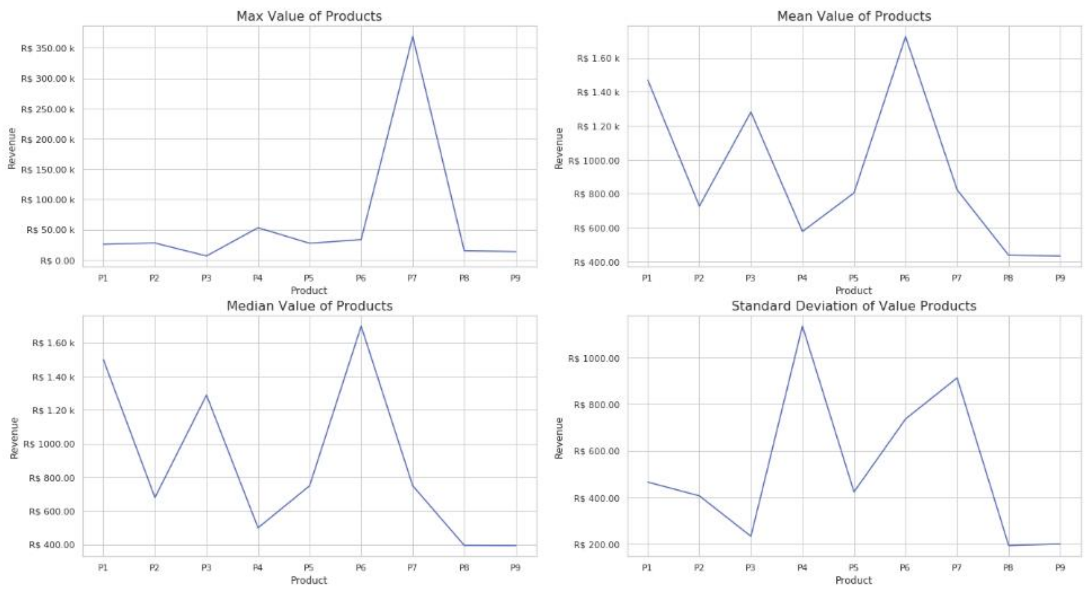
As well as in the analysis of quantity sold, I will show some measurements in the table below.



PROD_ID	min	max	mean	median	std	sum	count
P1	1145.78	25887.60	1470.079868	1499.00	466.296394	6014096.7	4091
P2	494.10	27907.00	726.308993	678.99	407.187108	46017485.2	63358
P3	1008.83	6588.00	1281.190048	1287.89	233.587447	3714170.0	2899
P4	265.81	53192.00	577.012720	499.00	1136.235288	8456698.4	14656
P5	561.75	27452.37	804.431663	749.00	424.021478	15907636.1	19775
P6	1133.19	33576.00	1725.870236	1697.89	737.648558	7079519.7	4102
P7	374.50	368750.00	822.920101	749.00	914.041828	161242141.7	195939
P8	229.00	15160.00	436.698291	394.90	193.924194	12157680.4	27840
P9	306.99	13795.60	432.906599	393.98	200.685378	7978901.5	18431

Figure 3

The value of 500 quantities sold of product P7 in an order, reflects in the data presented in the table above. The product P7 has the same behavior as in the previous table. For a better understanding, I will show below the data plotted in the chart.



In this scenario it’s possible to observe several behaviors. The first chart (Max Value) shows us the product P7 as the top 1 in the maximum value. But on the next chart we can see a completely scenario, the mean value of product P7 is significantly lower.

The product P6 has the highest mean value in related to other products. To explain this behavior we can see the sum and count columns. You can see that it has a high revenue value and a low frequency of sales per order.

4.3. Daily Revenue

The chart below shows total sales made from 01/01/2015 to 10/31/2015.

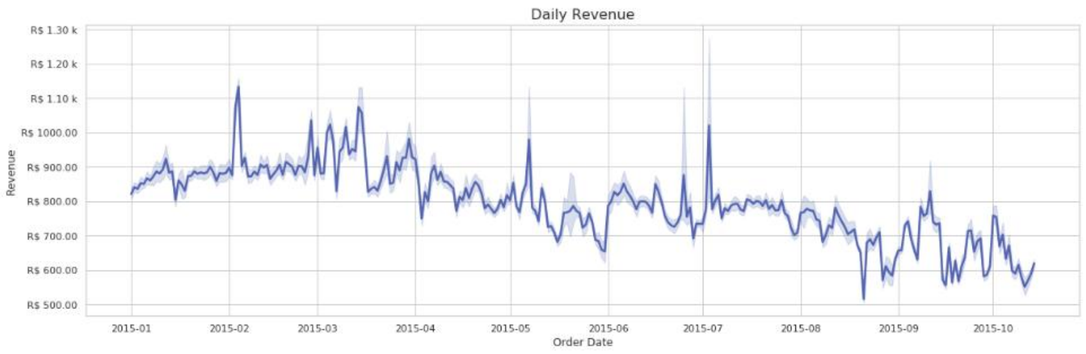


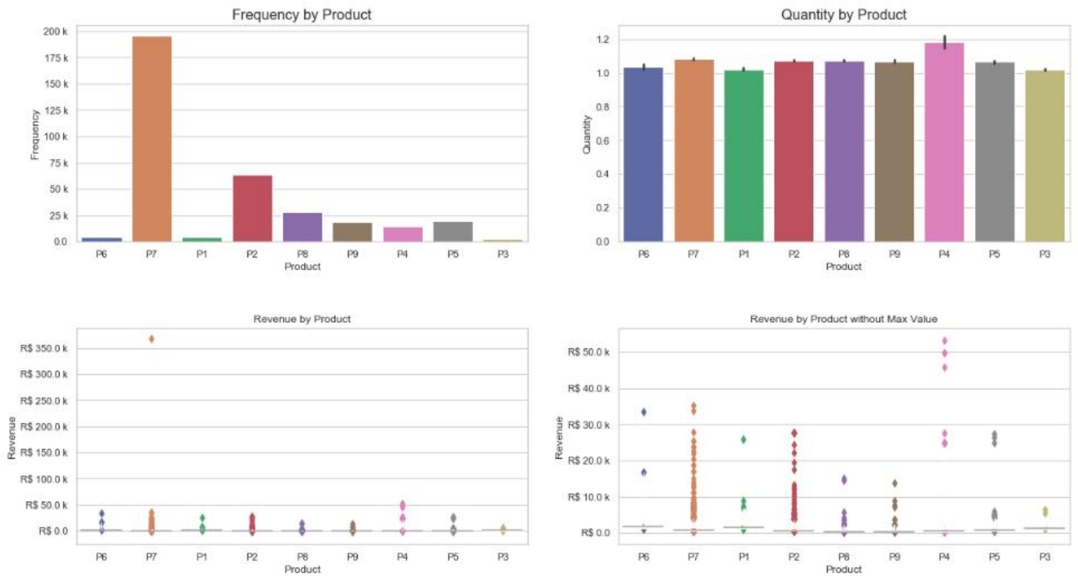
Figure 4

Sales during the 10-month period have several variations on specific days. When comparing these variations, it can be observed that most of them are peaks increasing in their total sold, few of them are frequent in the graph, even tending to decrease over time.

Another important factor to be observed are the positioning of each peak increase, much of it occurring at the beginning of the months.

4.4. Analysis of Products

The goal to be achieved in the data plot below is to understand the behavior of each product by analyzing its frequency, quantity and sales revenue.



Frequency by Product

In this plot we observed that the products 7 and 2 have a higher frequency in each sales in general (not being considered their respective quantities sold).

Quantity by Product

In this chart we analyze the total quantity sold of each product in each sale. Although products 7 and 2 have more frequencies in each sale, the data indicate that the product 4 has a larger quantity sold. Even though it is not as frequent in many sales, customers buy this product in larger quantities than other products.

Revenue by Product

As in the previous chart, product 7 continues to have a prominence in the value of revenue compared to other products. Here we clearly see that there is a sale where the value of your product is much higher than others, thus making this difference and the distance shown in the chart.

Revenue by Product without Max Value

In this plot we derive the significant revenue value from product 7 to better analyze the behavior of the other products. When we withdraw, we can first observe that product 4 has a significant revenue value right after product 7.

Even if product 4 stands out from the others in the chart, we can see that this difference depends on the quantity of products sold in each sale, since their values have a distance between them.

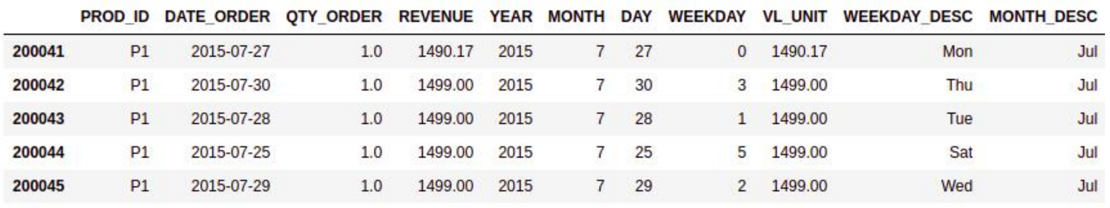
For product 7 we see a concentration very close to the points, leaving the line of its values more homogeneous than product 4, if we do not have a significant amount for the product, its value of prescription tends to decrease.

Product 2 also exhibits the same behavior of product 7 without the outlier, a line where its points (values) are closer to each other, thus having a greater conspicuity.

5. Data Preparation

This stage is responsible for preparing the data, where the dataframe will go through a data transformation process, so that it can be better used in the construction of the model.

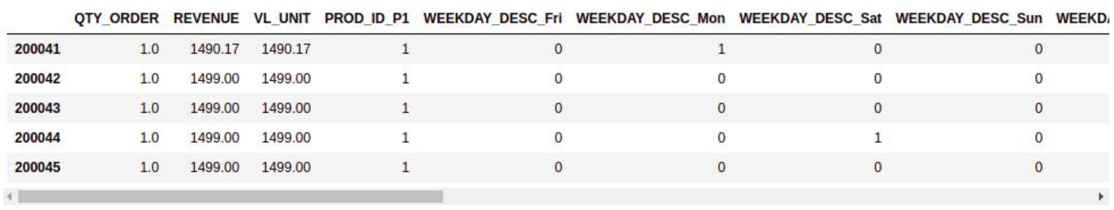
The data presented below, refer to product P1, but all other products follow the same structure.



	PROD_ID	DATE_ORDER	QTY_ORDER	REVENUE	YEAR	MONTH	DAY	WEEKDAY	VL_UNIT	WEEKDAY_DESC	MONTH_DESC
200041	P1	2015-07-27	1.0	1490.17	2015	7	27	0	1490.17	Mon	Jul
200042	P1	2015-07-30	1.0	1499.00	2015	7	30	3	1499.00	Thu	Jul
200043	P1	2015-07-28	1.0	1499.00	2015	7	28	1	1499.00	Tue	Jul
200044	P1	2015-07-25	1.0	1499.00	2015	7	25	5	1499.00	Sat	Jul
200045	P1	2015-07-29	1.0	1499.00	2015	7	29	2	1499.00	Wed	Jul

Figure 5

Since the models do not perform prediction with categorical data very well, the data transformation was necessary. Figure 6 shows the result of the transformation of the categorical data to dummy.



	QTY_ORDER	REVENUE	VL_UNIT	PROD_ID_P1	WEEKDAY_DESC_Fri	WEEKDAY_DESC_Mon	WEEKDAY_DESC_Sat	WEEKDAY_DESC_Sun	WEEKD
200041	1.0	1490.17	1490.17	1	0	1	0	0	
200042	1.0	1499.00	1499.00	1	0	0	0	0	
200043	1.0	1499.00	1499.00	1	0	0	0	0	
200044	1.0	1499.00	1499.00	1	0	0	1	0	
200045	1.0	1499.00	1499.00	1	0	0	0	0	

Figure 6

After the transformation of the data for each product, the analysis of the correlation of these features with respect to the QTY_ORDER response variable is necessary to know which of them can be used in modeling the model. Following is the heatmap graph of product P1.

P1

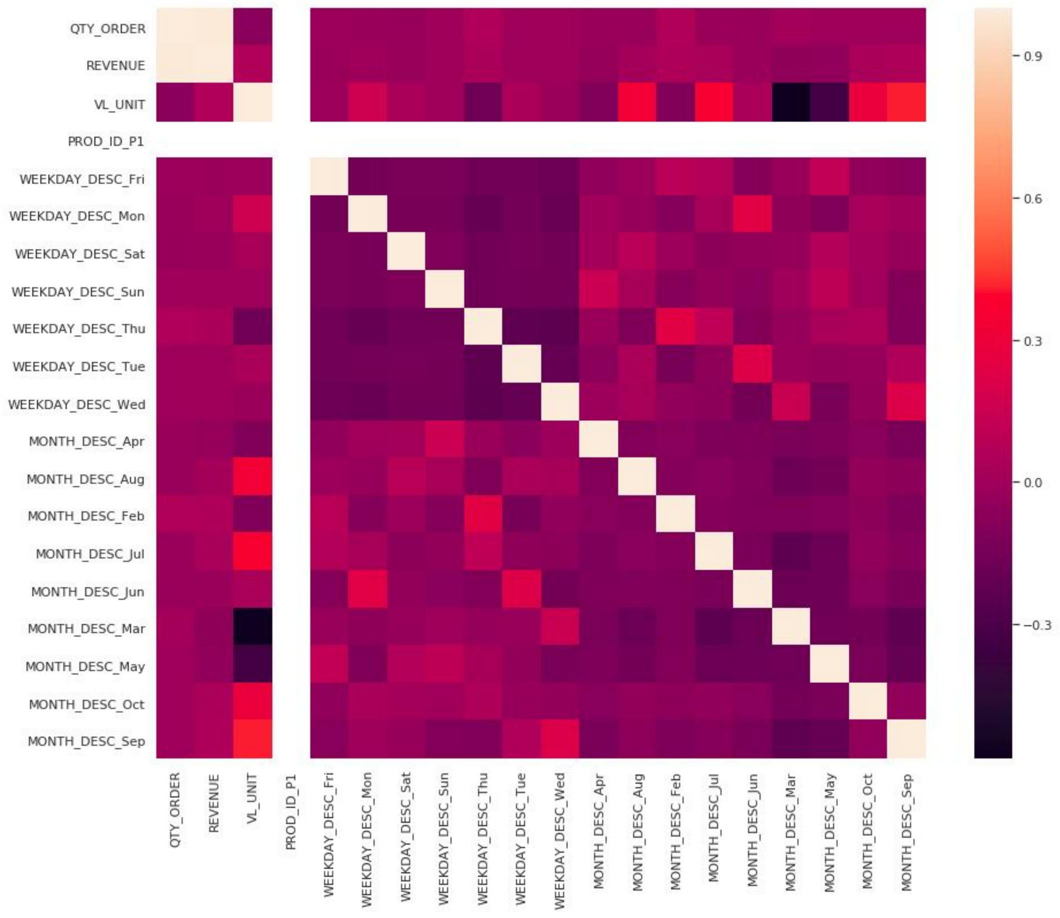


Figure 7 – Heatmap P1

This graph is indicating that few features correlate with the QTY_ORDER response variable, many variables are between 0.0 and 0.5.

P2

As in product P1, the dataframe structure of product P2 follows the same pattern as P1. What differs at this point are the correlations between the variables.

We can analyze the heatmap plotted below.

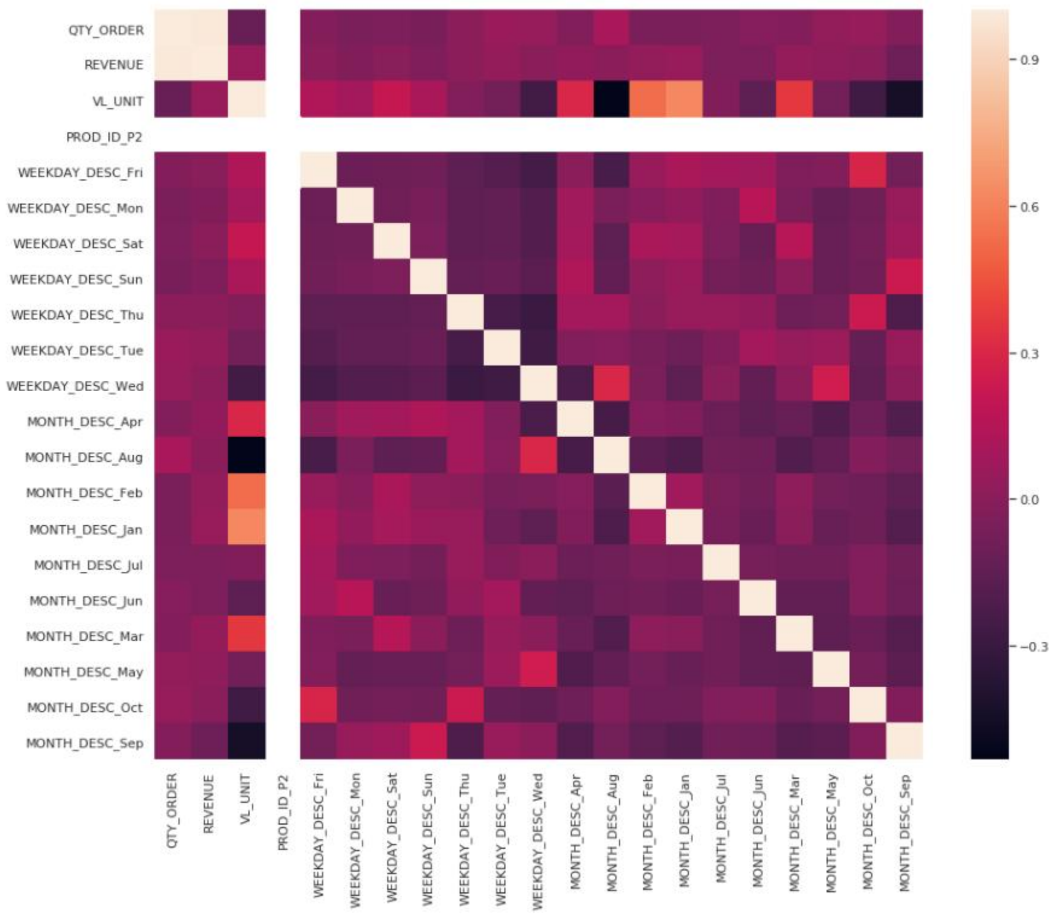


Figure 8 – Heatmap P2

Somente olhando para o gráfico, é possível analisar um comportamento semelhante ao gráfico do produto P1, porém no produto P2 existem mais áreas escuras indicando que existem mais grupos abaixo de -0.3.

P3

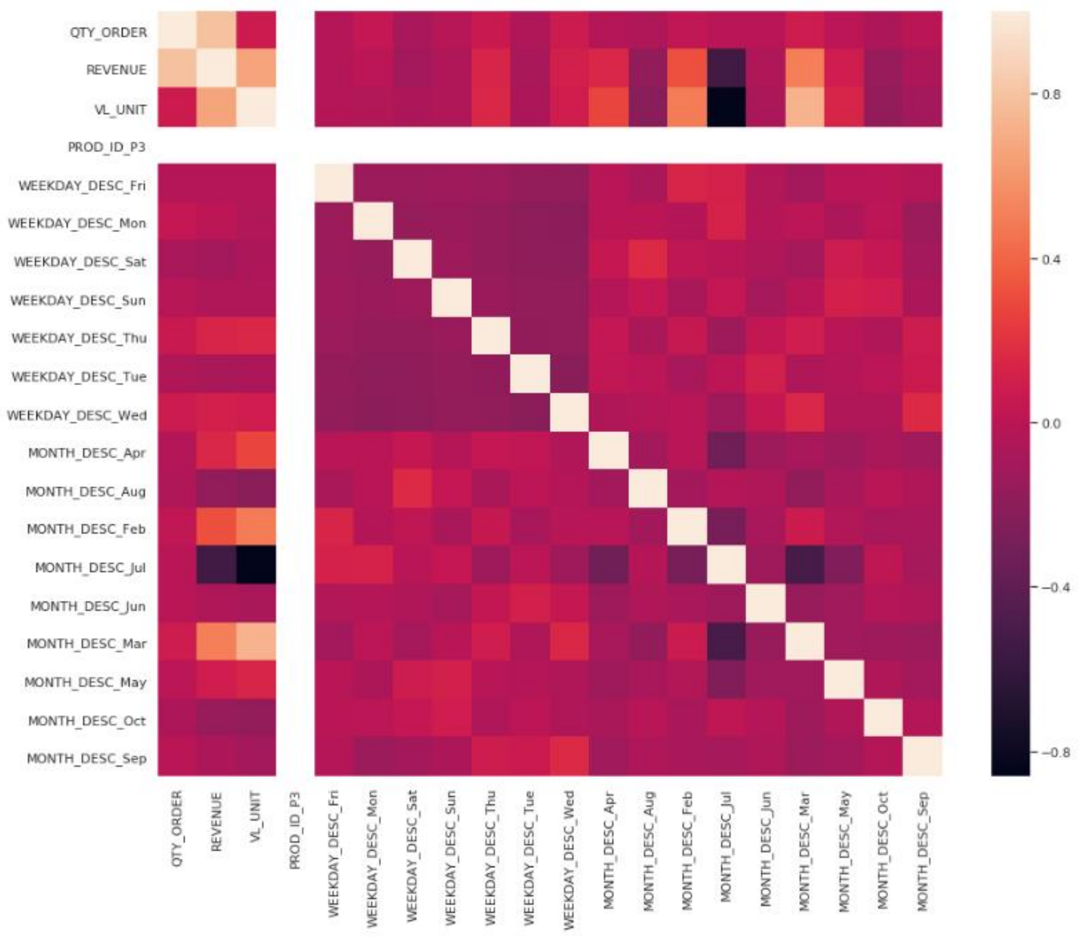


Figure 9 – Heatmap P3

Looking at the heatmap chart above, you can see a behavior equal to the graphics of the previous products.

P4

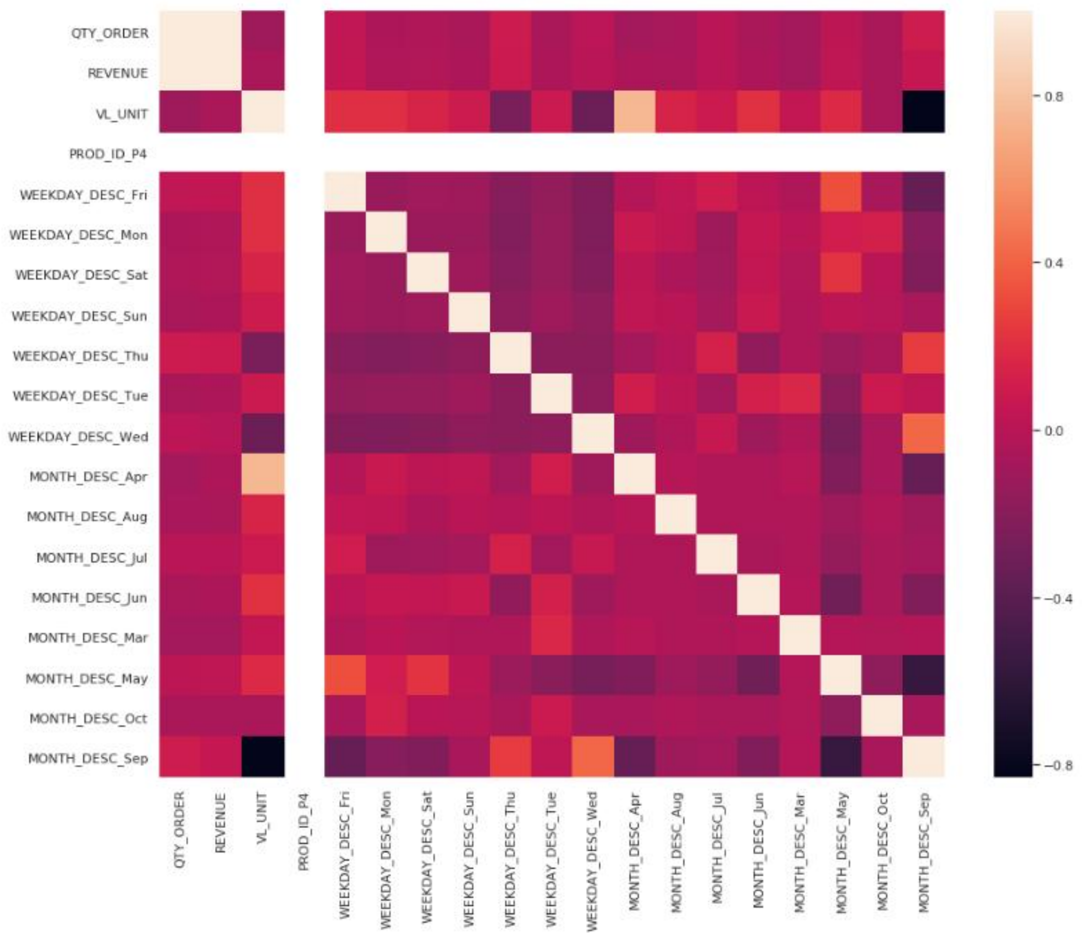


Figure 10 – Heatmap P4

By analyzing the line of the QTY_ORDER response variable, the graph tells us that the only significant variable for this model is REVENUE.

P5

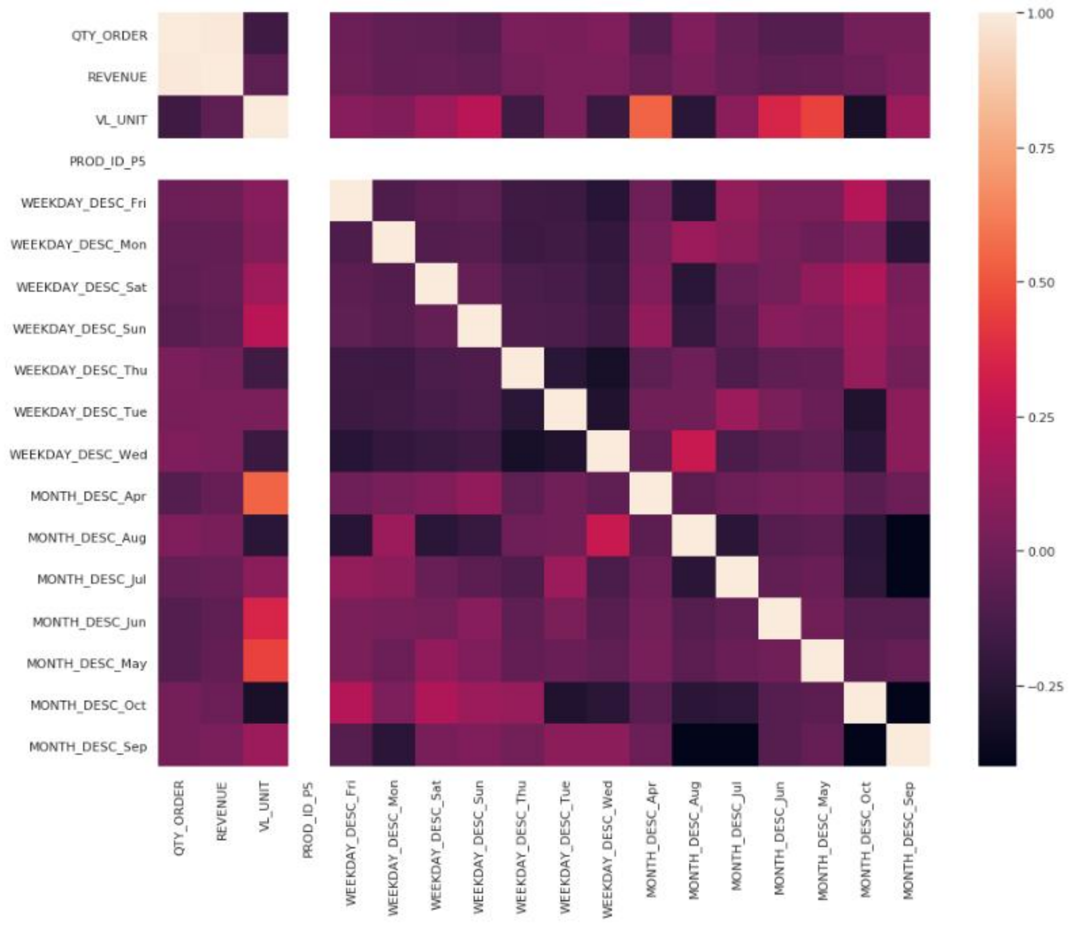


Figure 11 – Heatmap P5

In the product graph P5, it is observed that most of the values of the correlations are between 0.25 and 0.15, indicating that these variables are not significant to be used in the model.

P6

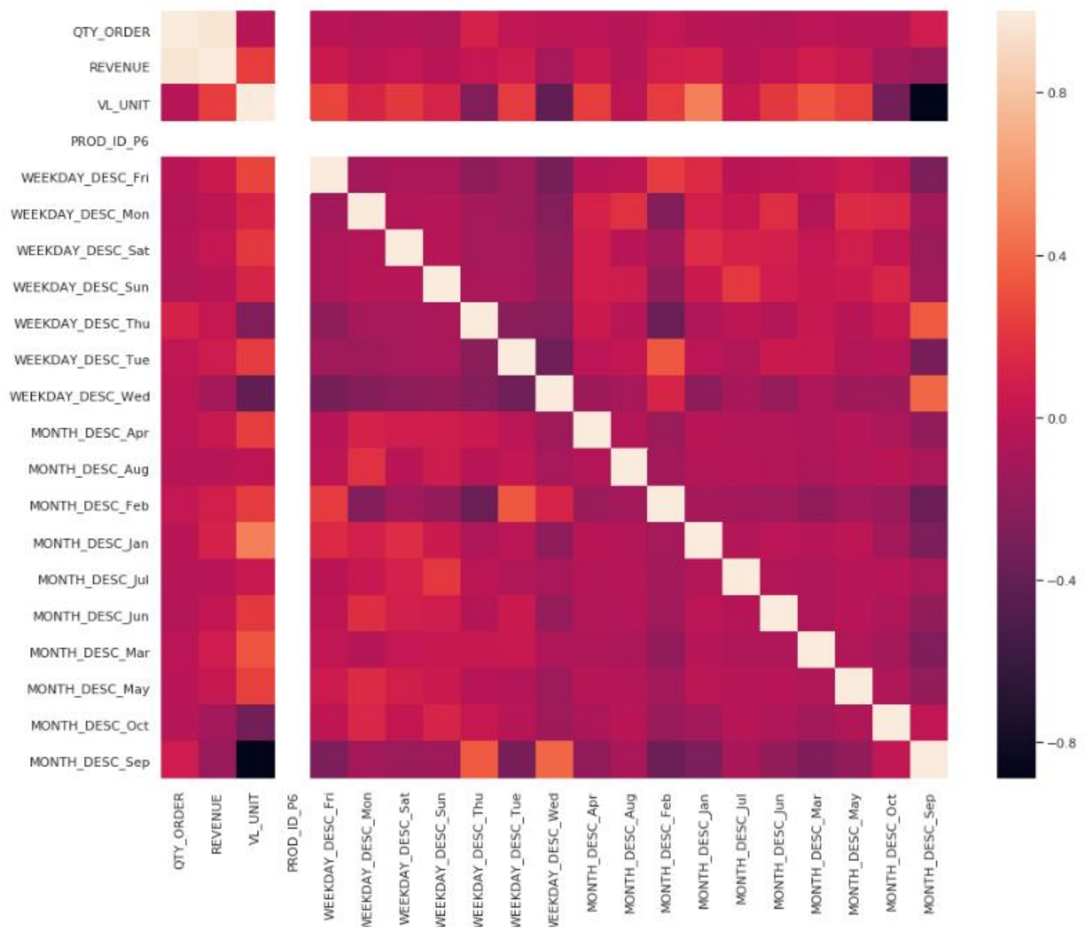


Figure 12 – Heatmap P6

The P6 product graph follows the same behavior as the others, only the variable REVENUE has a correlation with the response variable.

P7

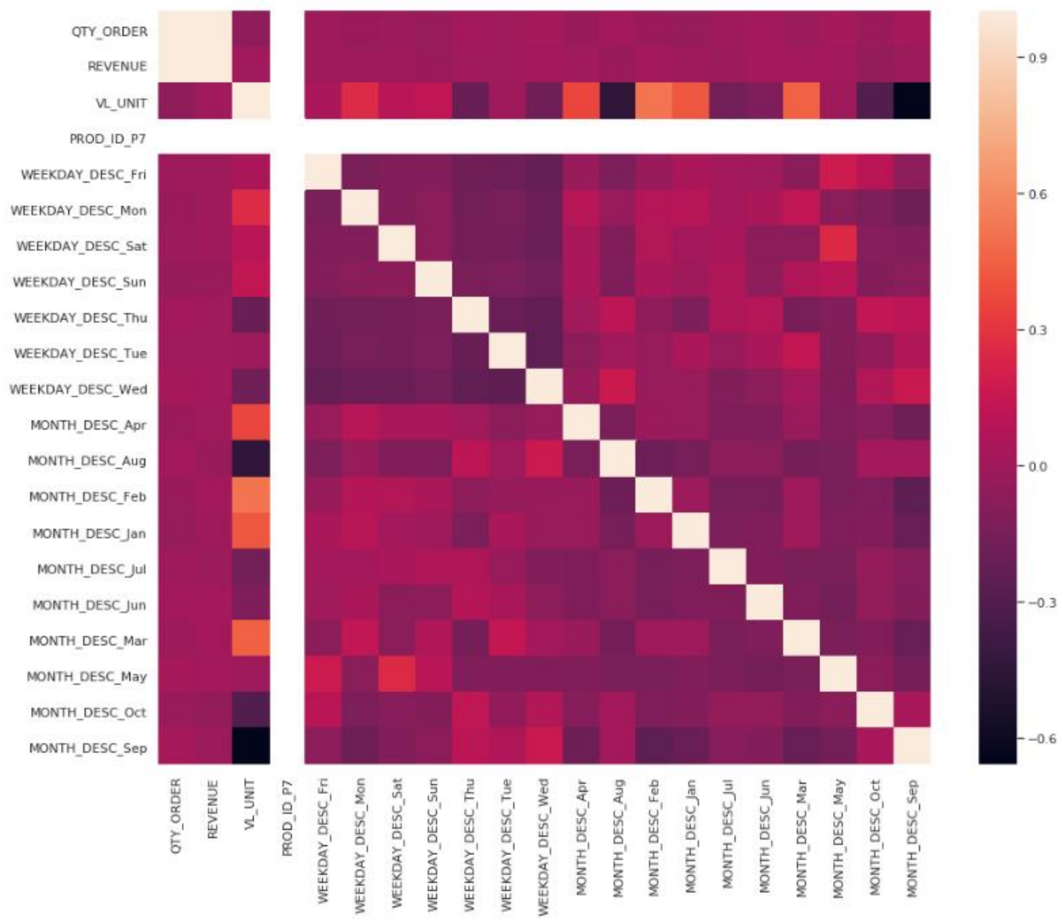


Figure 13 – Heatmap P7

The heatmap of the P7 product also behaves similarly to the correlation of the other products.

P8

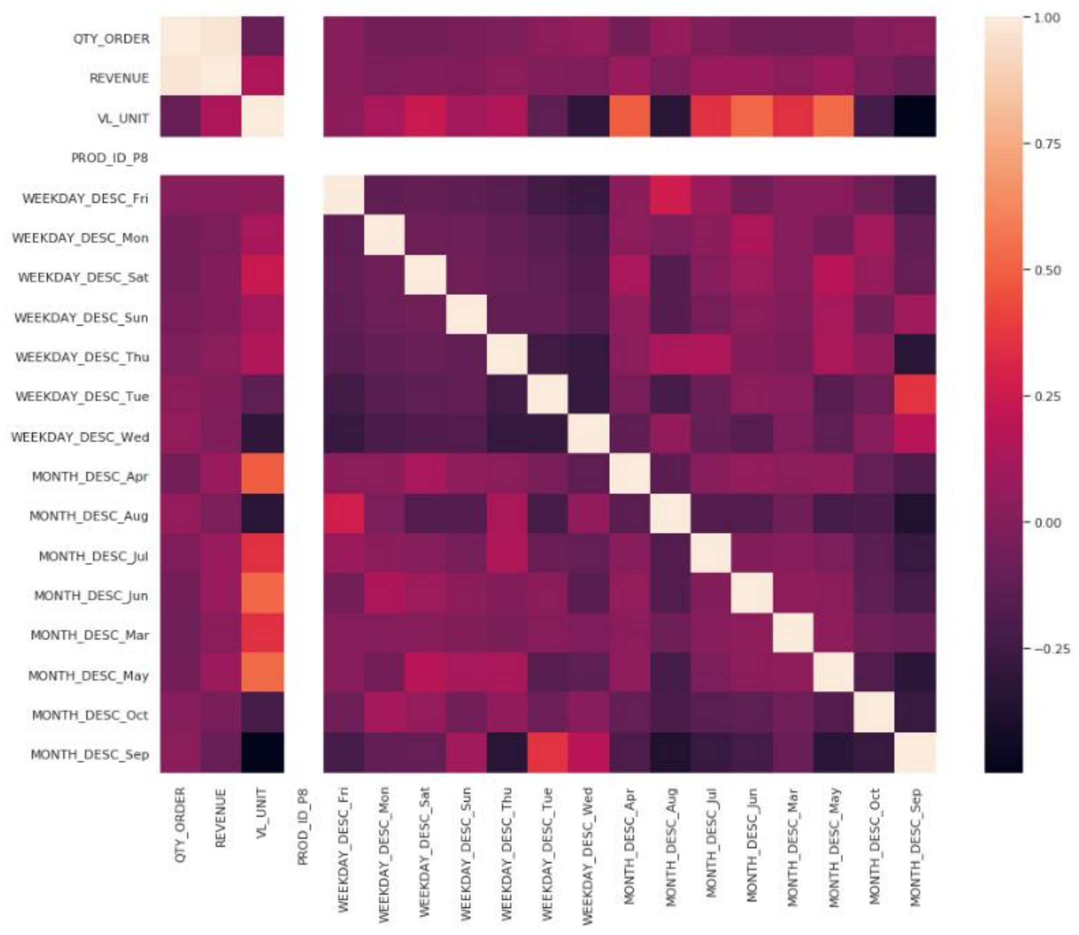


Figure 14 – Heatmap P8

Although the above heamap behaves similarly to other charts, their values are closer to 0.

P9

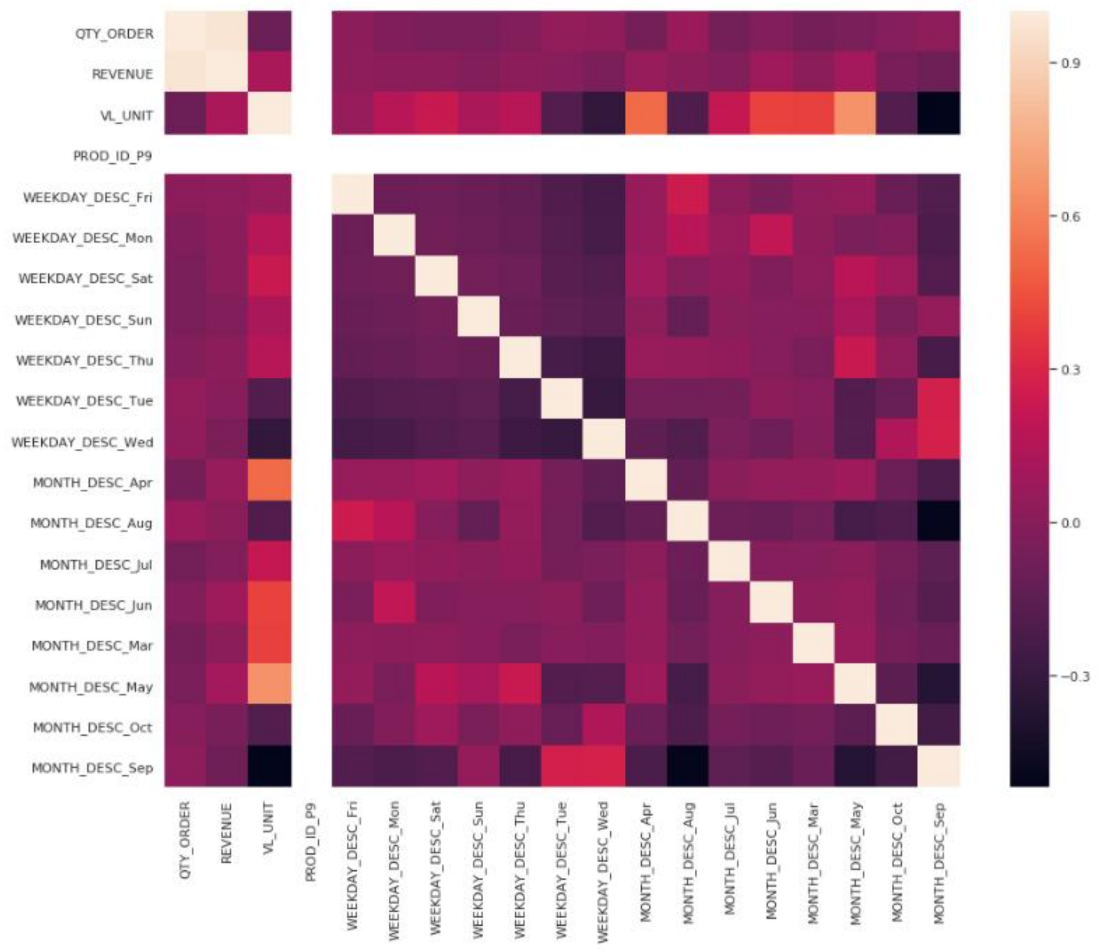


Figure 15 – Heatmap P9

The P9 product heatmap behaves similarly to other products.

6. Modeling

In this step I will do all the modeling part of the model, thus following the order of the previous processes. The methods applied to product P1 follow with default for the other products, so some images will not be included in this report.

P1

```
X = df_p1.drop('QTY_ORDER', axis=1)
y = df_p1['QTY_ORDER']
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)

X_train.shape, X_test.shape
```

```
((2863, 19), (1228, 19))
```

As part of the modeling process of the model, the database split was performed in 70% for the training base and 30% for the test base.

```
model_v1 = DecisionTreeRegressor(
    max_depth=4,
    min_samples_split=5,
    max_leaf_nodes=10
)

model_v1.fit(X_train, y_train)

for f, i in zip(X_train.columns, model_v1.feature_importances_):
    print('{:.<20}:{:3}'.format(f, i))
```

```
REVENUE.............:1.0
VL_UNIT.............:0.0
PROD_ID_P1..........:0.0
WEEKDAY_DESC_Fri....:0.0
WEEKDAY_DESC_Mon....:0.0
WEEKDAY_DESC_Sat....:0.0
WEEKDAY_DESC_Sun....:0.0
WEEKDAY_DESC_Thu....:0.0
WEEKDAY_DESC_Tue....:0.0
WEEKDAY_DESC_Wed....:0.0
MONTH_DESC_Apr......:0.0
MONTH_DESC_Aug......:0.0
MONTH_DESC_Feb......:0.0
MONTH_DESC_Jul......:0.0
MONTH_DESC_Jun......:0.0
MONTH_DESC_Mar......:0.0
MONTH_DESC_May......:0.0
MONTH_DESC_Oct......:0.0
MONTH_DESC_Sep......:0.0
```

In the image above, the model is adjusted with the decision tree algorithm, after which a for loop is performed to visualize the level of importance of each variable contained in the model.

Just as the heatmap graph indicates only the variable REVENUE, in this loop it is possible to numerically analyze the level of importance of these variables.

```
# Realizando um novo split nos dados com as variáveis significativas.
columns = ['REVENUE']

X_train, X_test, y_train, y_test = train_test_split(X[columns], y, test_size=0.3, random_state=0)

X_train.shape, X_test.shape
```

```
((2863, 1), (1228, 1))
```

```
print('{:10} {:20} {:20}'.format('depth', 'Training score', 'Testing score'))
print('{:10} {:20} {:20}'.format('-----', '--------------', '-------------'))

for i in range(10):
    score = compara_modelo(i, X_train, X_test, y_train, y_test)
    print('{:1}          {}                   {}'.format(i, score[0], score[1]))
```

```
depth      Training score       Testing score
-----      --------------       -------------
0          1.0                  0.961737793795374
1          0.8582019206803876                  -0.0013863549912849216
2          0.9599946103212348                  0.5617701982696841
3          1.0                  0.961737793795374
4          1.0                  0.961737793795374
5          1.0                  0.961737793795374
6          1.0                  0.961737793795374
7          1.0                  0.961737793795374
8          1.0                  0.961737793795374
9          1.0                  0.961737793795374
```

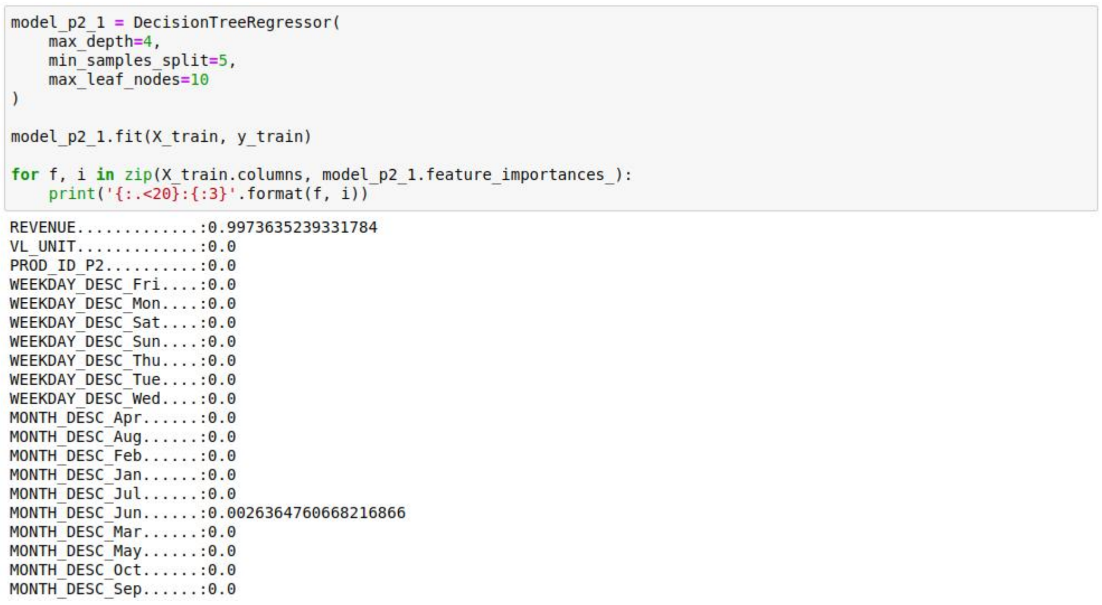
After visualizing the level of importance of the variables presented in the previous image, the split of the data is done again, but only applying the variable REVENUE.

With this step, a comparison of the score of the training and test base in the decision tree model is made, passing to the max_depth parameter a range of numbers from 1 to 10.

This is done to visualize if the model is overfitting and also indicate the ideal parameter to be used.

If the values between the training and test base are very different, it is concluded that the model is overfitting.

P2



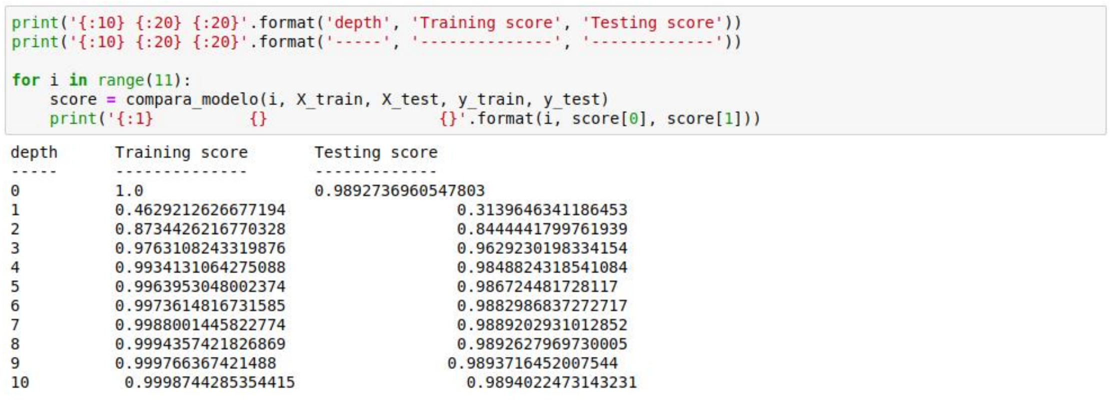
```
model_p2_1 = DecisionTreeRegressor(
    max_depth=4,
    min_samples_split=5,
    max_leaf_nodes=10
)

model_p2_1.fit(X_train, y_train)

for f, i in zip(X_train.columns, model_p2_1.feature_importances_):
    print('{:.<20}:{:3}'.format(f, i))
```

```
REVENUE.............:0.9973635239331784
VL_UNIT.............:0.0
PROD_ID_P2..........:0.0
WEEKDAY_DESC_Fri....:0.0
WEEKDAY_DESC_Mon....:0.0
WEEKDAY_DESC_Sat....:0.0
WEEKDAY_DESC_Sun....:0.0
WEEKDAY_DESC_Thu....:0.0
WEEKDAY_DESC_Tue....:0.0
WEEKDAY_DESC_Wed....:0.0
MONTH_DESC_Apr......:0.0
MONTH_DESC_Aug......:0.0
MONTH_DESC_Feb......:0.0
MONTH_DESC_Jan......:0.0
MONTH_DESC_Jul......:0.0
MONTH_DESC_Jun......:0.0026364760668216866
MONTH_DESC_Mar......:0.0
MONTH_DESC_May......:0.0
MONTH_DESC_Oct......:0.0
MONTH_DESC_Sep......:0.0
```

As in product P1, P2 behaves similarly to P1. Having only the variable REVENUE significant for the model.



```
print('{:10} {:20} {:20}'.format('depth', 'Training score', 'Testing score'))
print('{:10} {:20} {:20}'.format('-----', '--------------', '-------------'))

for i in range(11):
    score = compara_modelo(i, X_train, X_test, y_train, y_test)
    print('{:1}          {}                  {}'.format(i, score[0], score[1]))
```

```
depth      Training score       Testing score
-----      --------------       -------------
0          1.0                  0.9892736960547803
1          0.4629212626677194               0.3139646341186453
2          0.8734426216770328               0.8444441799761939
3          0.9763108243319876               0.9629230198334154
4          0.9934131064275088               0.9848824318541084
5          0.9963953048002374               0.986724481728117
6          0.9973614816731585               0.9882986837272717
7          0.9988001445822774               0.9889202931012852
8          0.9994357421826869               0.9892627969730005
9          0.999766367421488               0.9893716452007544
10          0.9998744285354415               0.9894022473143231
```

After resharing the data containing only the variable REVENUE, the comparison between the training and test data is shown in the figure above.

P3

```
model_p3_1 = DecisionTreeRegressor(
    max_depth=4,
    min_samples_split=5,
    max_leaf_nodes=10
)

model_p3_1.fit(X_train, y_train)

for f, i in zip(X_train.columns, model_p3_1.feature_importances_):
    print('{:.<20}:{:3}'.format(f, i))
```

```
REVENUE.............:1.0
VL_UNIT.............:0.0
PROD_ID_P3..........:0.0
WEEKDAY_DESC_Fri....:0.0
WEEKDAY_DESC_Mon....:0.0
WEEKDAY_DESC_Sat....:0.0
WEEKDAY_DESC_Sun....:0.0
WEEKDAY_DESC_Thu....:0.0
WEEKDAY_DESC_Tue....:0.0
WEEKDAY_DESC_Wed....:0.0
MONTH_DESC_Apr......:0.0
MONTH_DESC_Aug......:0.0
MONTH_DESC_Feb......:0.0
MONTH_DESC_Jul......:0.0
MONTH_DESC_Jun......:0.0
MONTH_DESC_Mar......:0.0
MONTH_DESC_May......:0.0
MONTH_DESC_Oct......:0.0
MONTH_DESC_Sep......:0.0
```

Just like in the other images, only the variable REVENUE has an importance level for the model.

```
print('{:10} {:20} {:20}'.format('depth', 'Training score', 'Testing score'))
print('{:10} {:20} {:20}'.format('-----', '--------------', '-------------'))

for i in range(10):
    score = compara_modelo(i, X_train, X_test, y_train, y_test)
    print('{:1}          {}                  {}'.format(i, score[0], score[1]))
```

```
depth      Training score       Testing score
-----      --------------       -------------
0          1.0                  0.9740832315529209
1          0.7842414518597226                  0.6108267507303247
2          0.9604525327853224                  0.9717610891000626
3          1.0                  0.9740832315529209
4          1.0                  0.9740832315529209
5          1.0                  0.9740832315529209
6          1.0                  0.9740832315529209
7          1.0                  0.9740832315529209
8          1.0                  0.9740832315529209
9          1.0                  0.9740832315529209
```

In parameter 1 the values of the training and test base begin to have a significant difference in relation to the other values, however as the value of the parameter increases the values become more constant.

P4

```
model_p4_1 = DecisionTreeRegressor(
    max_depth=4,
    min_samples_split=5,
    max_leaf_nodes=10
)

model_p4_1.fit(X_train, y_train)

for f, i in zip(X_train.columns, model_p4_1.feature_importances_):
    print('{:.<20}:{:3}'.format(f, i))
```

```
REVENUE.............:1.0
VL_UNIT.............:0.0
PROD_ID_P4..........:0.0
WEEKDAY_DESC_Fri....:0.0
WEEKDAY_DESC_Mon....:0.0
WEEKDAY_DESC_Sat....:0.0
WEEKDAY_DESC_Sun....:0.0
WEEKDAY_DESC_Thu....:0.0
WEEKDAY_DESC_Tue....:0.0
WEEKDAY_DESC_Wed....:0.0
MONTH_DESC_Apr......:0.0
MONTH_DESC_Aug......:0.0
MONTH_DESC_Jul......:0.0
MONTH_DESC_Jun......:0.0
MONTH_DESC_Mar......:0.0
MONTH_DESC_May......:0.0
MONTH_DESC_Oct......:0.0
MONTH_DESC_Sep......:0.0
```

Just like in the other images, only the variable REVENUE has an importance level for the model.

```
print('{:10} {:20} {:20}'.format('depth', 'Training score', 'Testing score'))
print('{:10} {:20} {:20}'.format('-----', '--------------', '-------------'))

for i in range(11):
    score = compara_modelo(i, X_train, X_test, y_train, y_test)
    print('{:1}          {}                  {}'.format(i, score[0], score[1]))
```

```
depth      Training score       Testing score
-----      --------------       -------------
0          1.0                  0.9934363118847349
1          0.7772464986732381               0.7706406079993345
2          0.9242261534954784               0.9410829185349286
3          0.9877815203934586               0.9850114919782668
4          0.9980157840367083               0.9930160903356664
5          0.9992849728363331               0.9932311262701466
6          0.9996101423104868               0.9933309133203878
7          0.9998426732528809               0.993399307037347
8          0.9999843074302864               0.9934331410209112
9          1.0                  0.9934363118847349
10          1.0                  0.9934363118847349
```

The values between the training and test database data are very similar.

P5

```
model_p5_1 = DecisionTreeRegressor(
    max_depth=4,
    min_samples_split=5,
    max_leaf_nodes=10
)

model_p5_1.fit(X_train, y_train)

for f, i in zip(X_train.columns, model_p5_1.feature_importances_):
    print('{:.<20}:{:3}'.format(f, i))
```

```
REVENUE.............:0.9959974248749486
VL_UNIT.............:0.004002575125051447
PROD_ID_P5..........:0.0
WEEKDAY_DESC_Fri....:0.0
WEEKDAY_DESC_Mon....:0.0
WEEKDAY_DESC_Sat....:0.0
WEEKDAY_DESC_Sun....:0.0
WEEKDAY_DESC_Thu....:0.0
WEEKDAY_DESC_Tue....:0.0
WEEKDAY_DESC_Wed....:0.0
MONTH_DESC_Apr......:0.0
MONTH_DESC_Aug......:0.0
MONTH_DESC_Jul......:0.0
MONTH_DESC_Jun......:0.0
MONTH_DESC_May......:0.0
MONTH_DESC_Oct......:0.0
MONTH_DESC_Sep......:0.0
```

Just like in the other images, only the variable REVENUE has an importance level for the model.

```
print('{:10} {:20} {:20}'.format('depth', 'Training score', 'Testing score'))
print('{:10} {:20} {:20}'.format('-----', '--------------', '-------------'))

for i in range(11):
    score = compara_modelo(i, X_train, X_test, y_train, y_test)
    print('{:1}          {}                  {}'.format(i, score[0], score[1]))
```

```
depth      Training score       Testing score
-----      --------------       -------------
0          1.0                  0.9936678416620279
1          0.5021550729906958               0.3458651946248169
2          0.9793415322994393               0.9836374917770825
3          0.9928386263448529               0.9912729263608759
4          0.9955945828796501               0.9927211355985246
5          0.9973453771244243               0.9923205746194764
6          0.9987409678406036               0.9929309290563108
7          0.9995852107028943               0.9933542927588191
8          0.9996773861022512               0.9934384156352898
9          1.0                  0.9936678416620279
10          1.0                  0.9936678416620279
```

Except the value parameter 1, the values between the training and test database data are very similar.

P6

```python
model_p6_1 = DecisionTreeRegressor(
    max_depth=4,
    min_samples_split=5,
    max_leaf_nodes=10
)

model_p6_1.fit(X_train, y_train)

for f, i in zip(X_train.columns, model_p6_1.feature_importances_):
    print('{:.<20}:{:3}'.format(f, i))
```

```
REVENUE.............:0.9996043021494209
VL_UNIT.............:0.00039569785057918914
PROD_ID_P6..........:0.0
WEEKDAY_DESC_Fri....:0.0
WEEKDAY_DESC_Mon....:0.0
WEEKDAY_DESC_Sat....:0.0
WEEKDAY_DESC_Sun....:0.0
WEEKDAY_DESC_Thu....:0.0
WEEKDAY_DESC_Tue....:0.0
WEEKDAY_DESC_Wed....:0.0
MONTH_DESC_Apr......:0.0
MONTH_DESC_Aug......:0.0
MONTH_DESC_Feb......:0.0
MONTH_DESC_Jan......:0.0
MONTH_DESC_Jul......:0.0
MONTH_DESC_Jun......:0.0
MONTH_DESC_Mar......:0.0
MONTH_DESC_May......:0.0
MONTH_DESC_Oct......:0.0
MONTH_DESC_Sep......:0.0
```

Just like in the other images, only the variable REVENUE has an importance level for the model.

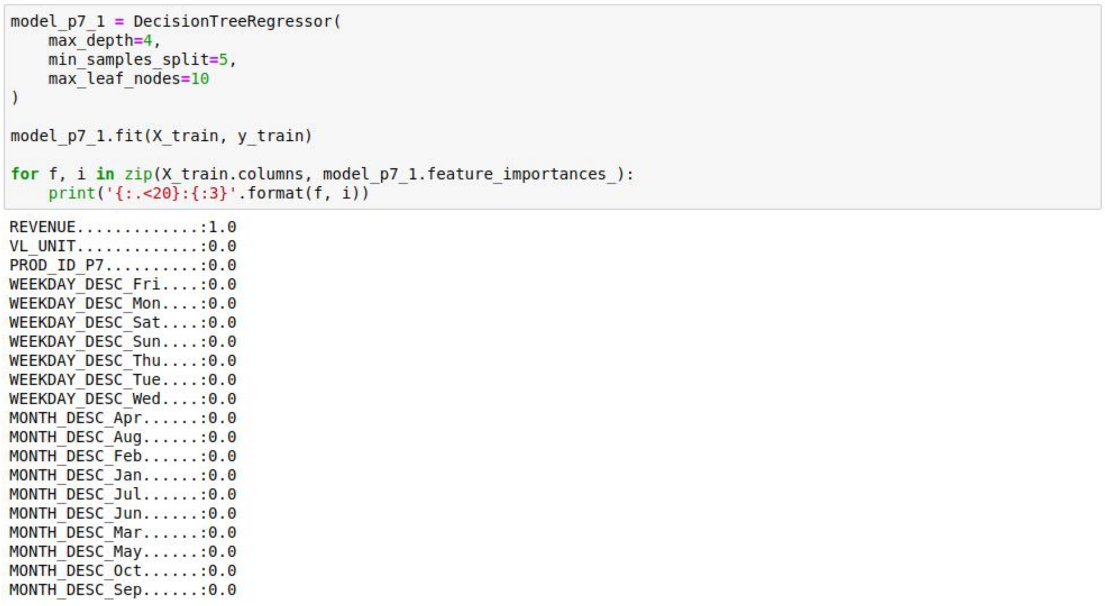
```python
print('{:10} {:20} {:20}'.format('depth', 'Training score', 'Testing score'))
print('{:10} {:20} {:20}'.format('-----', '--------------', '-------------'))

for i in range(11):
    score = compara_modelo(i, X_train, X_test, y_train, y_test)
    print('{:1}          {}                  {}'.format(i, score[0], score[1]))
```

```
depth      Training score       Testing score
-----      --------------       -------------
0          1.0                  0.9538640281837943
1          0.7509274379709331                  -0.0014367191558384285
2          0.9607350641844321                  0.8533260104608122
3          0.994972054446752                  0.9397517593557273
4          0.9991376874380973                  0.8885047347775028
5          0.9993101499504778                  0.9128542754582781
6          1.0                  0.9538640281837943
7          1.0                  0.9538640281837943
8          1.0                  0.9538640281837943
9          1.0                  0.9538640281837943
10          1.0                  0.9538640281837943
```

From parameter 3, the values between the training and test database data become very similar.

P7



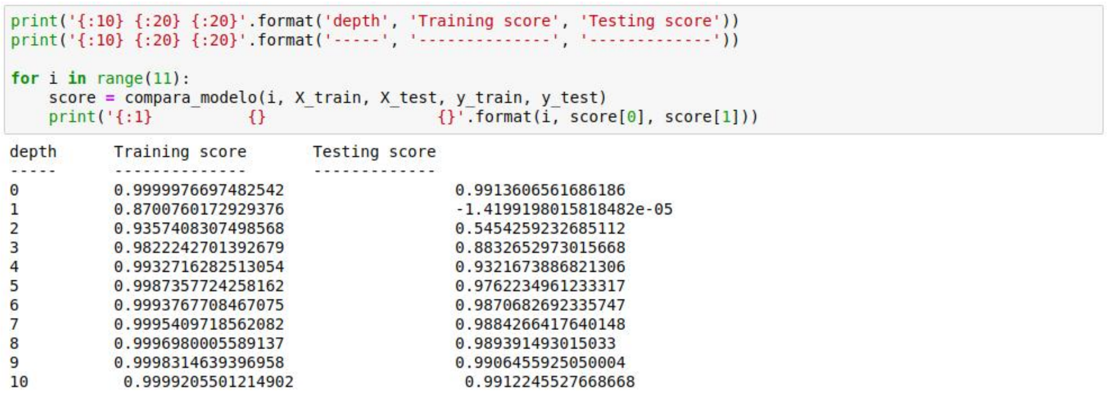
```
model_p7_1 = DecisionTreeRegressor(
    max_depth=4,
    min_samples_split=5,
    max_leaf_nodes=10
)

model_p7_1.fit(X_train, y_train)

for f, i in zip(X_train.columns, model_p7_1.feature_importances_):
    print('{:.<20}:{:3}'.format(f, i))
```

```
REVENUE.............:1.0
VL_UNIT.............:0.0
PROD_ID_P7..........:0.0
WEEKDAY_DESC_Fri....:0.0
WEEKDAY_DESC_Mon....:0.0
WEEKDAY_DESC_Sat....:0.0
WEEKDAY_DESC_Sun....:0.0
WEEKDAY_DESC_Thu....:0.0
WEEKDAY_DESC_Tue....:0.0
WEEKDAY_DESC_Wed....:0.0
MONTH_DESC_Apr......:0.0
MONTH_DESC_Aug......:0.0
MONTH_DESC_Feb......:0.0
MONTH_DESC_Jan......:0.0
MONTH_DESC_Jul......:0.0
MONTH_DESC_Jun......:0.0
MONTH_DESC_Mar......:0.0
MONTH_DESC_May......:0.0
MONTH_DESC_Oct......:0.0
MONTH_DESC_Sep......:0.0
```

Just like in the other images, only the variable REVENUE has an importance level for the model.



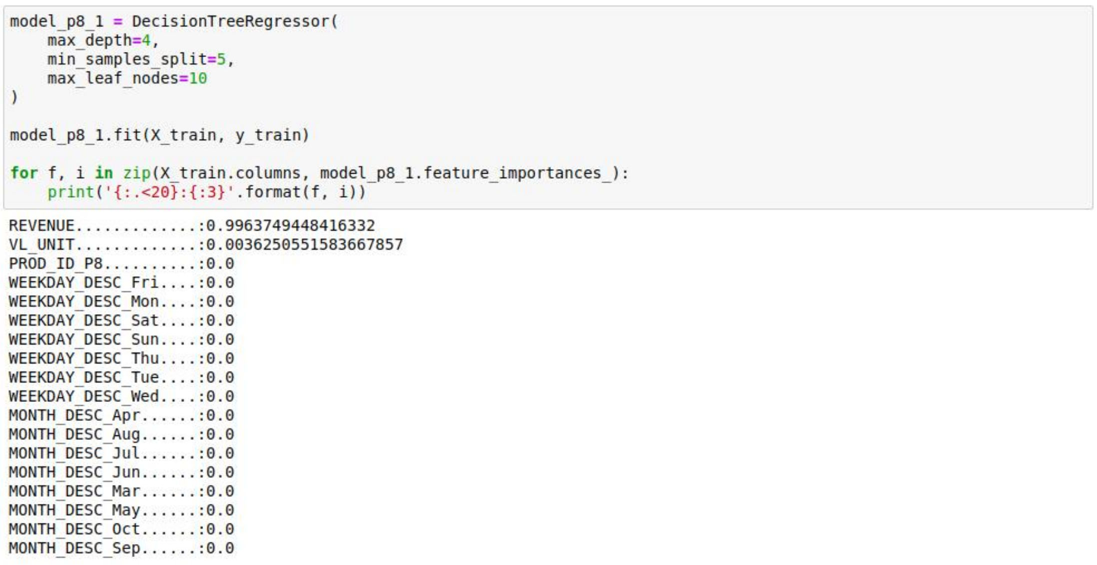
```
print('{:10} {:20} {:20}'.format('depth', 'Training score', 'Testing score'))
print('{:10} {:20} {:20}'.format('-----', '--------------', '-------------'))

for i in range(11):
    score = compara_modelo(i, X_train, X_test, y_train, y_test)
    print('{:1}          {}                    {}'.format(i, score[0], score[1]))
```

```
depth      Training score       Testing score
-----      --------------       -------------
0          0.9999976697482542              0.9913606561686186
1          0.8700760172929376              -1.4199198015818482e-05
2          0.9357408307498568              0.5454259232685112
3          0.9822242701392679              0.8832652973015668
4          0.9932716282513054              0.9321673886821306
5          0.9987357724258162              0.9762234961233317
6          0.9993767708467075              0.9870682692335747
7          0.9995409718562082              0.9884266417640148
8          0.9996980005589137              0.989391493015033
9          0.9998314639396958              0.9906455925050004
10          0.9999205501214902              0.9912245527668668
```

From parameter 4, the values between the training and test database data are very similar.

P8



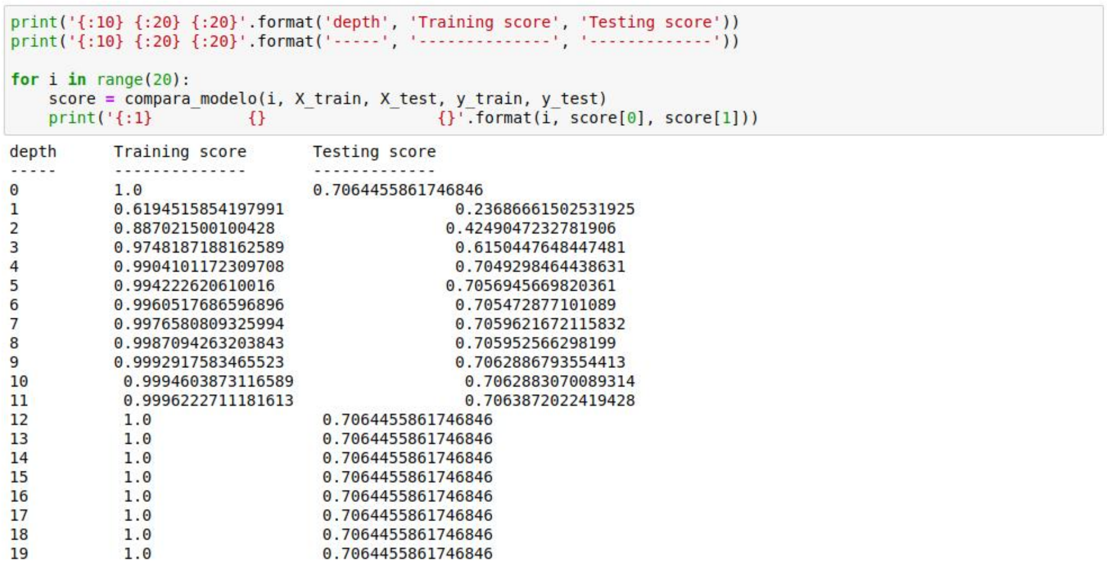
```
model_p8_1 = DecisionTreeRegressor(
    max_depth=4,
    min_samples_split=5,
    max_leaf_nodes=10
)

model_p8_1.fit(X_train, y_train)

for f, i in zip(X_train.columns, model_p8_1.feature_importances_):
    print('{:.<20}:{:3}'.format(f, i))
```

```
REVENUE.............:0.9963749448416332
VL_UNIT.............:0.0036250551583667857
PROD_ID_P8..........:0.0
WEEKDAY_DESC_Fri....:0.0
WEEKDAY_DESC_Mon....:0.0
WEEKDAY_DESC_Sat....:0.0
WEEKDAY_DESC_Sun....:0.0
WEEKDAY_DESC_Thu....:0.0
WEEKDAY_DESC_Tue....:0.0
WEEKDAY_DESC_Wed....:0.0
MONTH_DESC_Apr......:0.0
MONTH_DESC_Aug......:0.0
MONTH_DESC_Jul......:0.0
MONTH_DESC_Jun......:0.0
MONTH_DESC_Mar......:0.0
MONTH_DESC_May......:0.0
MONTH_DESC_Oct......:0.0
MONTH_DESC_Sep......:0.0
```

Just like in the other images, only the variable REVENUE has an importance level for the model.



```
print('{:10} {:20} {:20}'.format('depth', 'Training score', 'Testing score'))
print('{:10} {:20} {:20}'.format('-----', '--------------', '-------------'))

for i in range(20):
    score = compara_modelo(i, X_train, X_test, y_train, y_test)
    print('{:1}          {}                   {}'.format(i, score[0], score[1]))
```

```
depth      Training score       Testing score
-----      --------------       -------------
0          1.0                  0.7064455861746846
1          0.6194515854197991              0.23686661502531925
2          0.887021500100428              0.4249047232781906
3          0.9748187188162589              0.6150447648447481
4          0.9904101172309708              0.7049298464438631
5          0.994222620610016              0.7056945669820361
6          0.9960517686596896              0.705472877101089
7          0.9976580809325994              0.7059621672115832
8          0.9987094263203843              0.705952566298199
9          0.9992917583465523              0.7062886793554413
10          0.9994603873116589              0.7062883070089314
11          0.9996222711181613              0.7063872022419428
12          1.0                  0.7064455861746846
13          1.0                  0.7064455861746846
14          1.0                  0.7064455861746846
15          1.0                  0.7064455861746846
16          1.0                  0.7064455861746846
17          1.0                  0.7064455861746846
18          1.0                  0.7064455861746846
19          1.0                  0.7064455861746846
```

For this model, the data between the evaluation of the training and test data has a lot of variation, featuring overfitting.

To correct this it was necessary to apply the log method to leave the range of values more closer.

```
model_p8_1 = DecisionTreeRegressor(
    random_state=0,
    max_depth=4,
    max_leaf_nodes=20,
    min_samples_leaf=20,
    min_samples_split=10
)
model_p8_1.fit(X_train, np.log(y_train))

train_score = dt.score(X_train, np.log(y_train))
test_score = dt.score(X_test, np.log(y_test))

y_pred = model_p8_2.predict(X_test)

print(train_score, test_score)
print('\nRMSE: {}'.format(mean_squared_error(y_test, y_pred)**0.5))
```

```
0.9946097215603933 0.9774998664683455

RMSE: 0.14378662021572808
```

P9

```
model_p9_1 = DecisionTreeRegressor(
    max_depth=4,
    min_samples_split=5,
    max_leaf_nodes=10
)

model_p9_1.fit(X_train, y_train)

for f, i in zip(X_train.columns, model_p9_1.feature_importances_):
    print('{:.<20}:{:3}'.format(f, i))
```

```
REVENUE.............:0.9982094076227004
VL_UNIT.............:0.0017905923772996676
PROD_ID_P9..........:0.0
WEEKDAY_DESC_Fri....:0.0
WEEKDAY_DESC_Mon....:0.0
WEEKDAY_DESC_Sat....:0.0
WEEKDAY_DESC_Sun....:0.0
WEEKDAY_DESC_Thu....:0.0
WEEKDAY_DESC_Tue....:0.0
WEEKDAY_DESC_Wed....:0.0
MONTH_DESC_Apr......:0.0
MONTH_DESC_Aug......:0.0
MONTH_DESC_Jul......:0.0
MONTH_DESC_Jun......:0.0
MONTH_DESC_Mar......:0.0
MONTH_DESC_May......:0.0
MONTH_DESC_Oct......:0.0
MONTH_DESC_Sep......:0.0
```

Just like in the other images, only the variable REVENUE has an importance level for the model.

```
print('{:10} {:20} {:20}'.format('depth', 'Training score', 'Testing score'))
print('{:10} {:20} {:20}'.format('-----', '--------------', '-------------'))

for i in range(11):
    score = compara_modelo(i, X_train, X_test, y_train, y_test)
    print('{:1}          {}                {}'.format(i, score[0], score[1]))
```

```
depth      Training score       Testing score
-----      --------------       -------------
0          1.0                  0.9921423266670203
1          0.4813010636437708               -1.9911374843584184e-05
2          0.8469868462311068               0.6160819625457199
3          0.9735216420676012               0.9347330305493862
4          0.9921438477284393               0.9766781348037576
5          0.9978016069575787               0.9975260482131473
6          0.9988333711445753               0.9958452652952541
7          0.9995685366879464               0.994616542145571
8          0.9997707851154716               0.9934161291799838
9          0.9997860661077734               0.9933700881252984
10          1.0                  0.9921423266670203
```

From parameter 3, the values between the training and test base data are very similar.

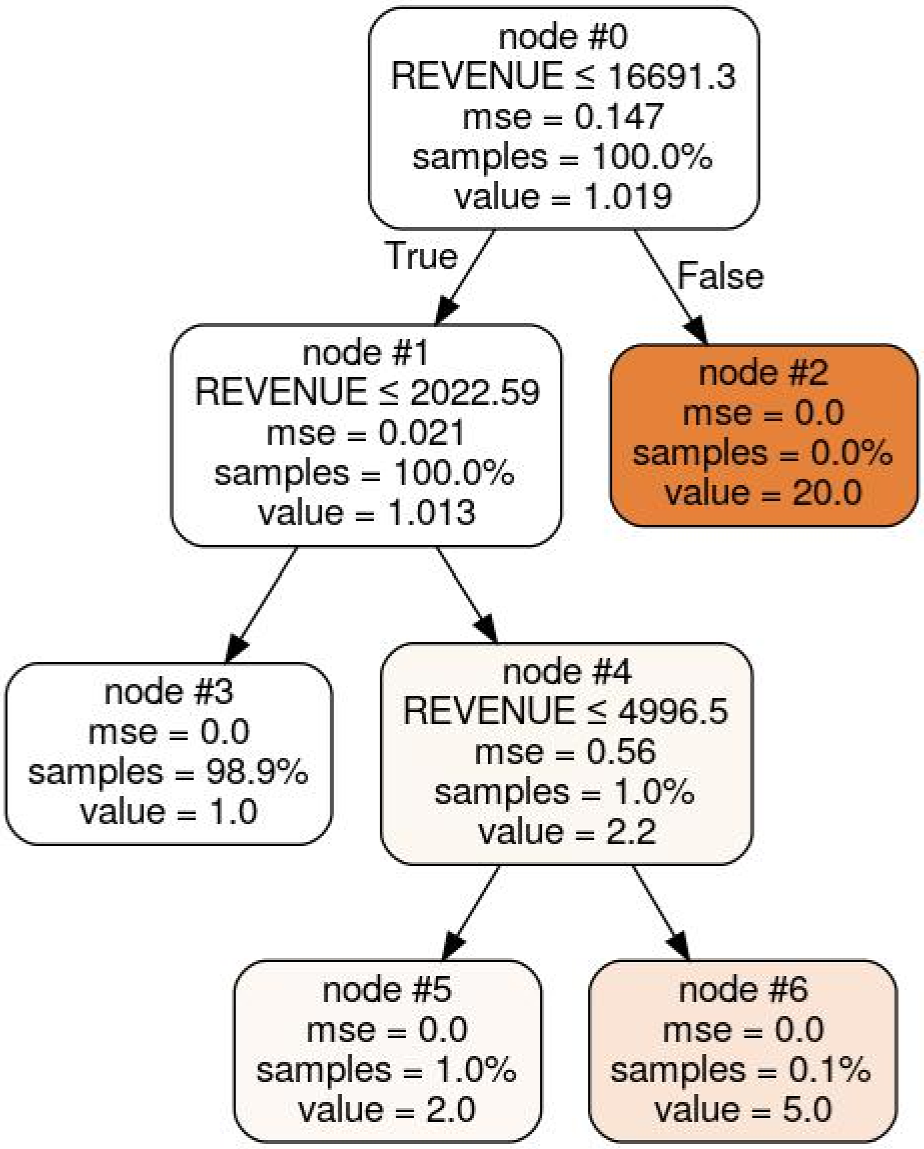
7. Evaluation

Summary of the results of the RMSE (Root Mean Square Error) method.

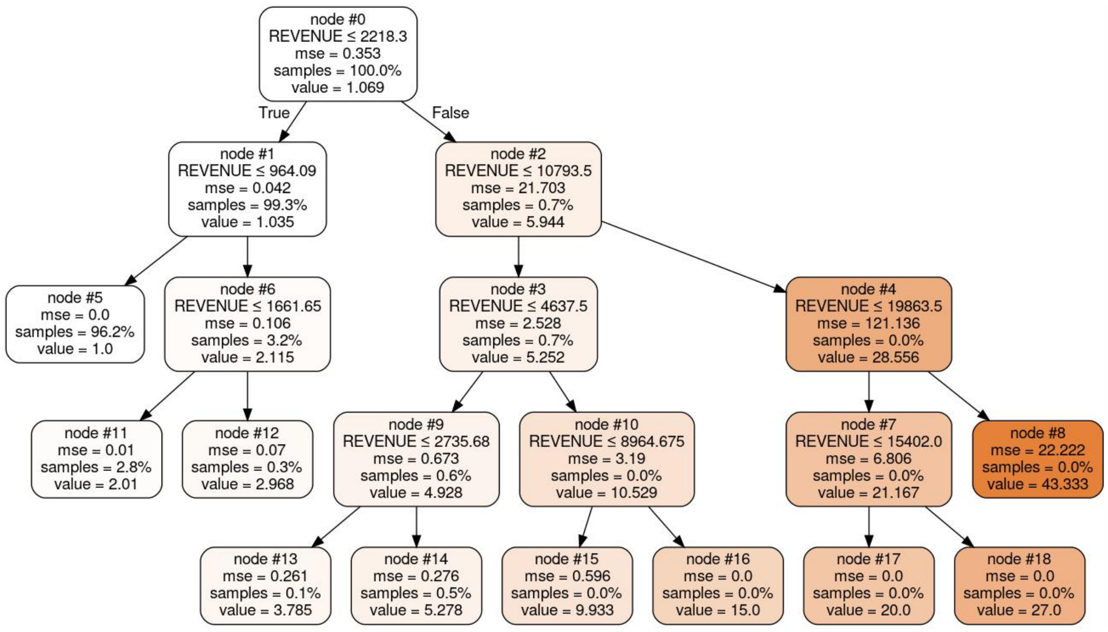
	P1	P2	P3	P4	P5	P6	P7	P8	P9
RMSE	0.049	0.123	0.033	0.300	0.064	0.300	0.070	0.442	0.022

The image of the decision tree applied to the model of each product will be listed below. In it it is possible to observe which paths the model chose as criterion.

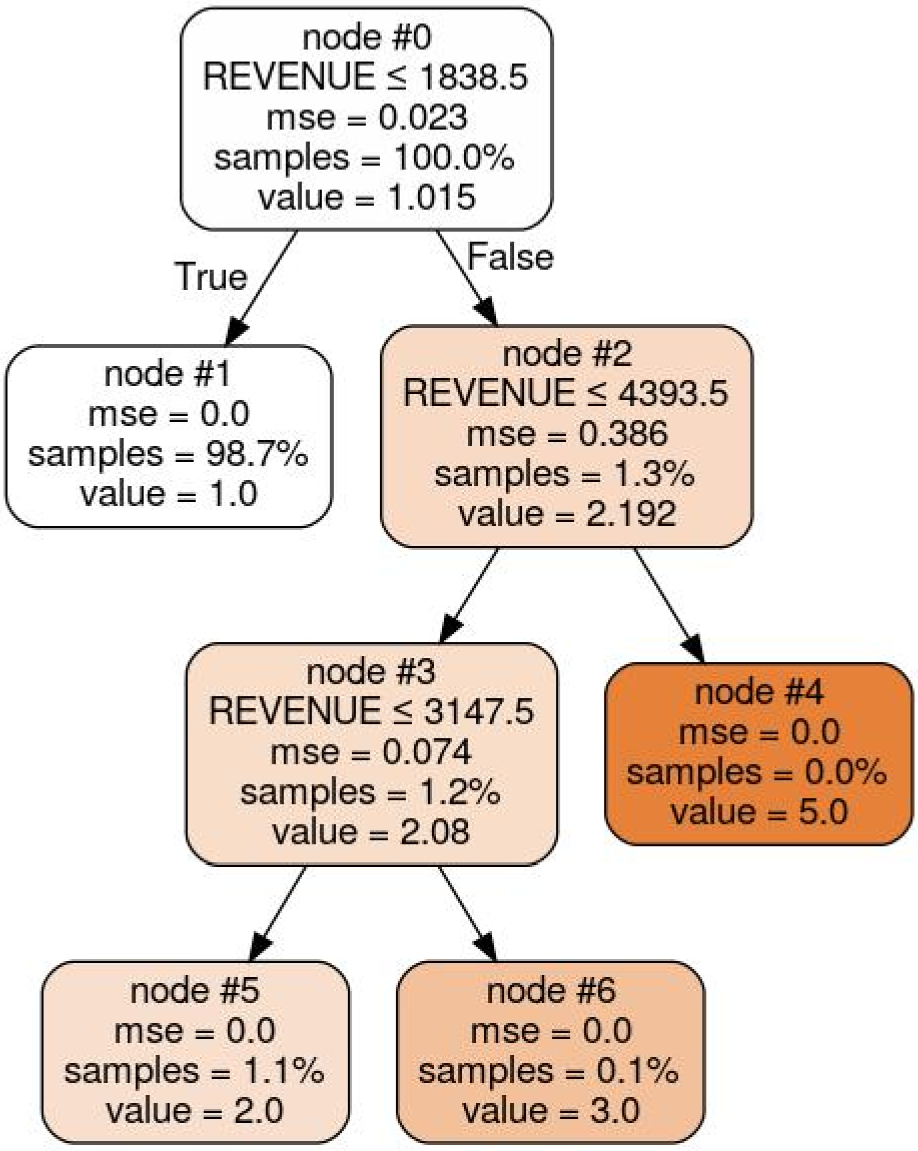
P1



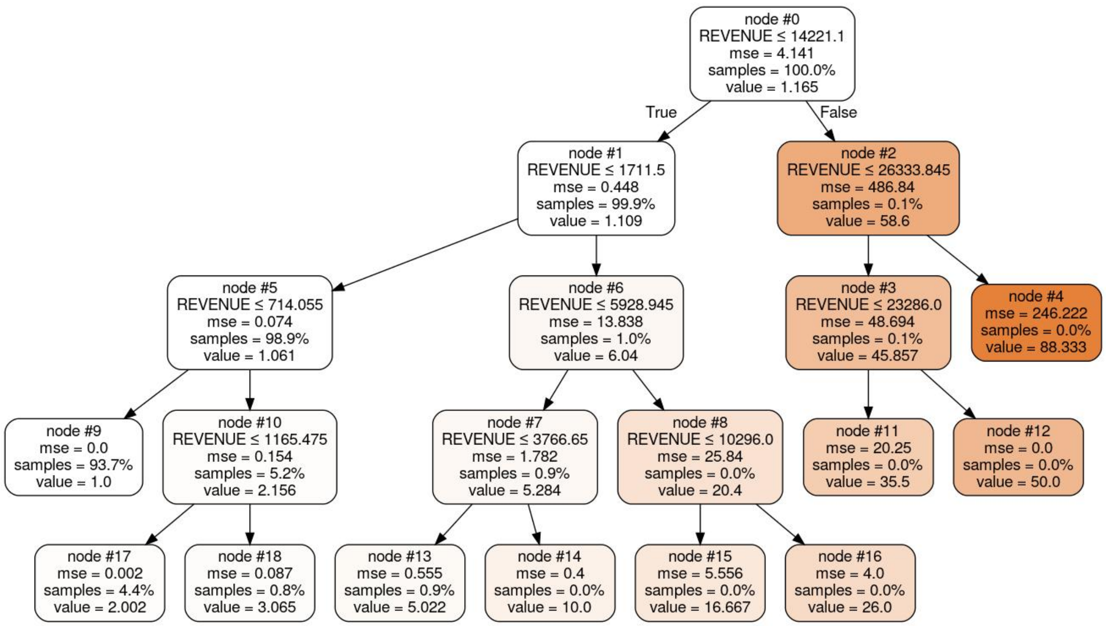
P2


node #0
REVENUE ≤ 2218.3
mse = 0.353
samples = 100.0%
value = 1.069
True
False
node #1
REVENUE ≤ 964.09
mse = 0.042
samples = 99.3%
value = 1.035
node #2
REVENUE ≤ 10793.5
mse = 21.703
samples = 0.7%
value = 5.944
node #5
mse = 0.0
samples = 96.2%
value = 1.0
node #6
REVENUE ≤ 1661.65
mse = 0.106
samples = 3.2%
value = 2.115
node #3
REVENUE ≤ 4637.5
mse = 2.528
samples = 0.7%
value = 5.252
node #4
REVENUE ≤ 19863.5
mse = 121.136
samples = 0.0%
value = 28.556
node #11
mse = 0.01
samples = 2.8%
value = 2.01
node #12
mse = 0.07
samples = 0.3%
value = 2.968
node #9
REVENUE ≤ 2735.68
mse = 0.673
samples = 0.6%
value = 4.928
node #10
REVENUE ≤ 8964.675
mse = 3.19
samples = 0.0%
value = 10.529
node #7
REVENUE ≤ 15402.0
mse = 6.806
samples = 0.0%
value = 21.167
node #8
mse = 22.222
samples = 0.0%
value = 43.333
node #13
mse = 0.261
samples = 0.1%
value = 3.785
node #14
mse = 0.276
samples = 0.5%
value = 5.278
node #15
mse = 0.596
samples = 0.0%
value = 9.933
node #16
mse = 0.0
samples = 0.0%
value = 15.0
node #17
mse = 0.0
samples = 0.0%
value = 20.0
node #18
mse = 0.0
samples = 0.0%
value = 27.0

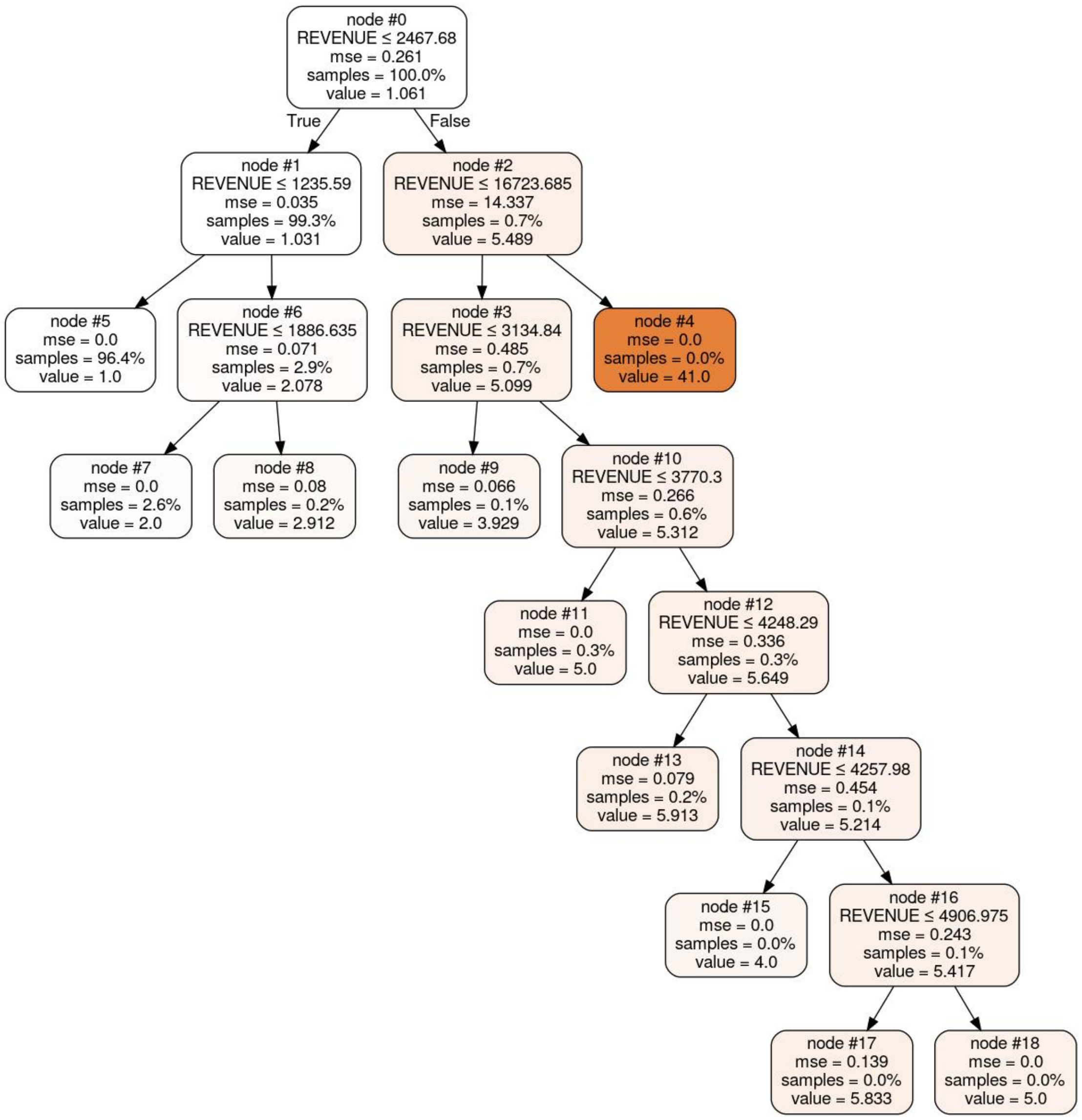
P3


node #0
REVENUE ≤ 1838.5
mse = 0.023
samples = 100.0%
value = 1.015
True
False
node #1
mse = 0.0
samples = 98.7%
value = 1.0
node #2
REVENUE ≤ 4393.5
mse = 0.386
samples = 1.3%
value = 2.192
node #3
REVENUE ≤ 3147.5
mse = 0.074
samples = 1.2%
value = 2.08
node #4
mse = 0.0
samples = 0.0%
value = 5.0
node #5
mse = 0.0
samples = 1.1%
value = 2.0
node #6
mse = 0.0
samples = 0.1%
value = 3.0

P4

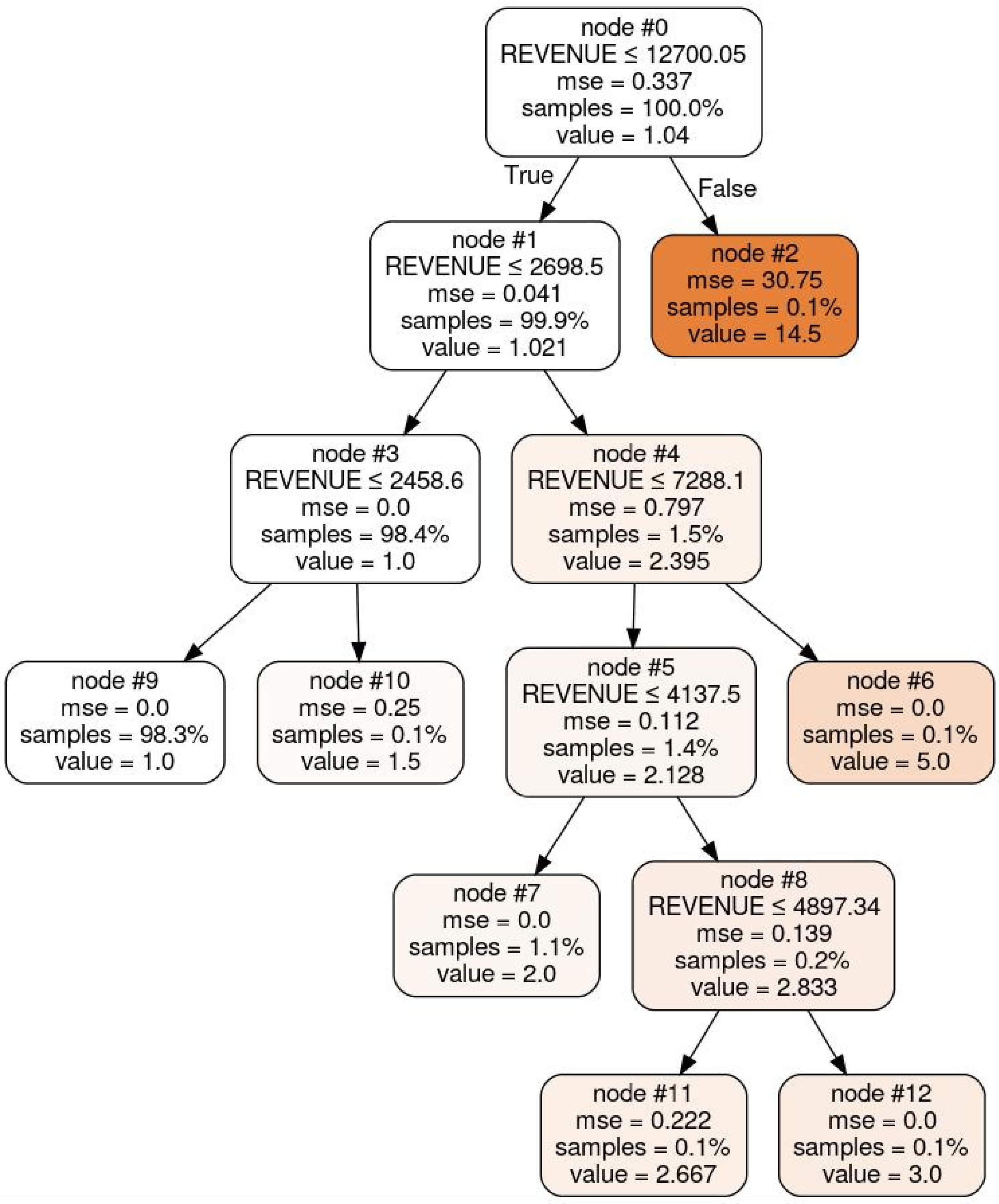

node #0
REVENUE ≤ 14221.1
mse = 4.141
samples = 100.0%
value = 1.165
True
False
node #1
REVENUE ≤ 1711.5
mse = 0.448
samples = 99.9%
value = 1.109
node #2
REVENUE ≤ 26333.845
mse = 486.84
samples = 0.1%
value = 58.6
node #5
REVENUE ≤ 714.055
mse = 0.074
samples = 98.9%
value = 1.061
node #6
REVENUE ≤ 5928.945
mse = 13.838
samples = 1.0%
value = 6.04
node #3
REVENUE ≤ 23286.0
mse = 48.694
samples = 0.1%
value = 45.857
node #4
mse = 246.222
samples = 0.0%
value = 88.333
node #9
mse = 0.0
samples = 93.7%
value = 1.0
node #10
REVENUE ≤ 1165.475
mse = 0.154
samples = 5.2%
value = 2.156
node #7
REVENUE ≤ 3766.65
mse = 1.782
samples = 0.9%
value = 5.284
node #8
REVENUE ≤ 10296.0
mse = 25.84
samples = 0.0%
value = 20.4
node #11
mse = 20.25
samples = 0.0%
value = 35.5
node #12
mse = 0.0
samples = 0.0%
value = 50.0
node #17
mse = 0.002
samples = 4.4%
value = 2.002
node #18
mse = 0.087
samples = 0.8%
value = 3.065
node #13
mse = 0.555
samples = 0.9%
value = 5.022
node #14
mse = 0.4
samples = 0.0%
value = 10.0
node #15
mse = 5.556
samples = 0.0%
value = 16.667
node #16
mse = 4.0
samples = 0.0%
value = 26.0

P5

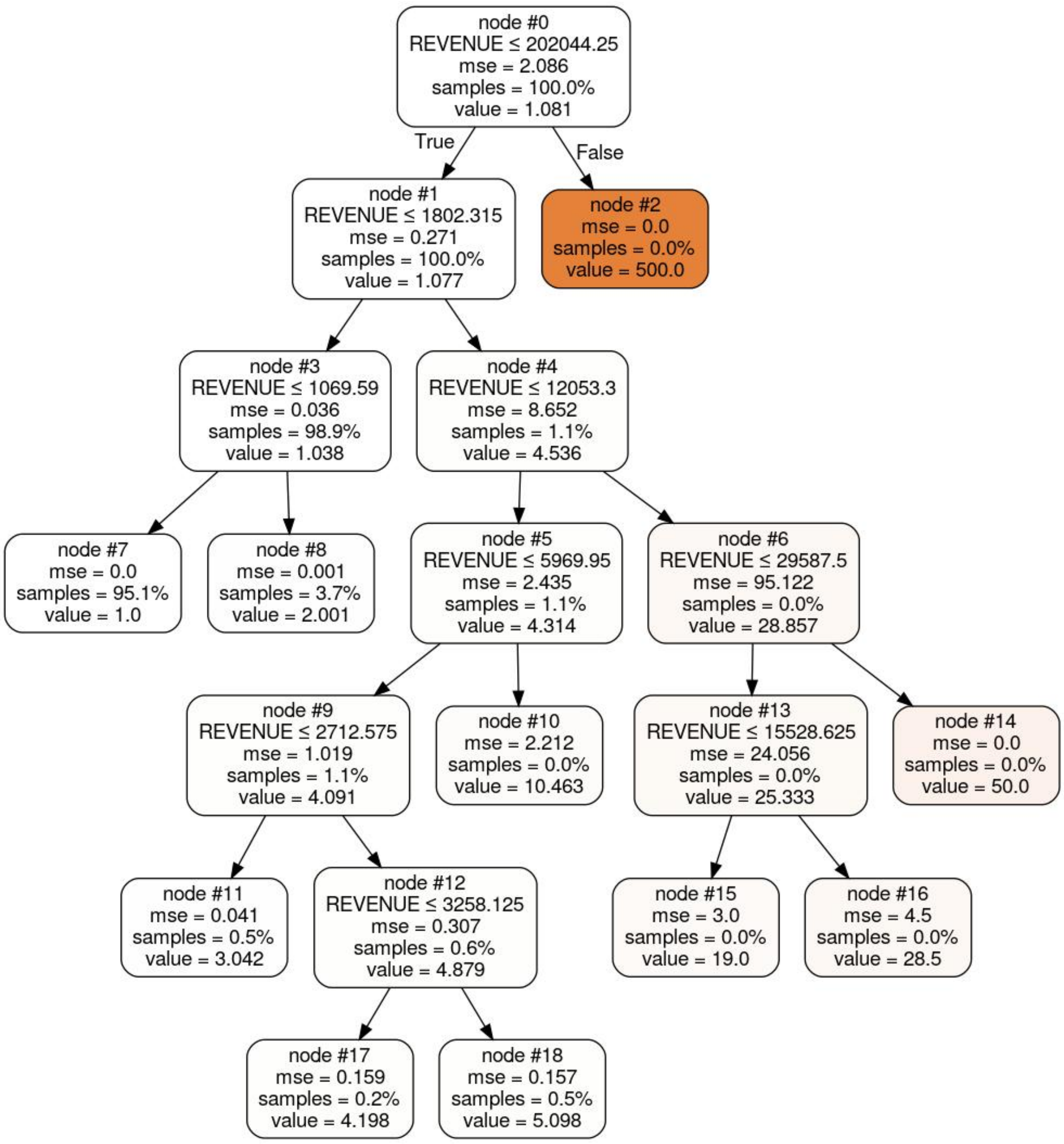


node #0
REVENUE ≤ 2467.68
mse = 0.261
samples = 100.0%
value = 1.061
True
False
node #1
REVENUE ≤ 1235.59
mse = 0.035
samples = 99.3%
value = 1.031
node #2
REVENUE ≤ 16723.685
mse = 14.337
samples = 0.7%
value = 5.489
node #5
mse = 0.0
samples = 96.4%
value = 1.0
node #6
REVENUE ≤ 1886.635
mse = 0.071
samples = 2.9%
value = 2.078
node #3
REVENUE ≤ 3134.84
mse = 0.485
samples = 0.7%
value = 5.099
node #4
mse = 0.0
samples = 0.0%
value = 41.0
node #7
mse = 0.0
samples = 2.6%
value = 2.0
node #8
mse = 0.08
samples = 0.2%
value = 2.912
node #9
mse = 0.066
samples = 0.1%
value = 3.929
node #10
REVENUE ≤ 3770.3
mse = 0.266
samples = 0.6%
value = 5.312
node #11
mse = 0.0
samples = 0.3%
value = 5.0
node #12
REVENUE ≤ 4248.29
mse = 0.336
samples = 0.3%
value = 5.649
node #13
mse = 0.079
samples = 0.2%
value = 5.913
node #14
REVENUE ≤ 4257.98
mse = 0.454
samples = 0.1%
value = 5.214
node #15
mse = 0.0
samples = 0.0%
value = 4.0
node #16
REVENUE ≤ 4906.975
mse = 0.243
samples = 0.1%
value = 5.417
node #17
mse = 0.139
samples = 0.0%
value = 5.833
node #18
mse = 0.0
samples = 0.0%
value = 5.0

P6

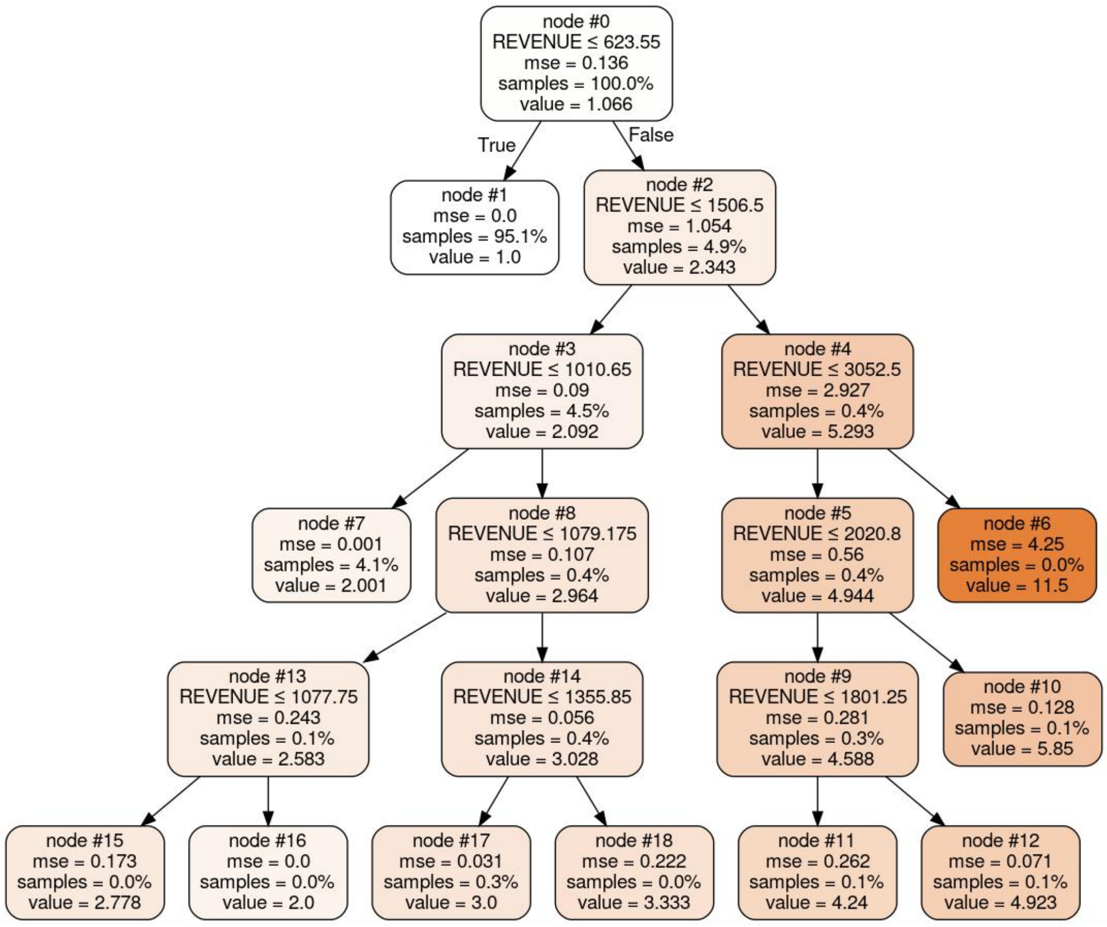

node #0
REVENUE ≤ 12700.05
mse = 0.337
samples = 100.0%
value = 1.04
True
False
node #1
REVENUE ≤ 2698.5
mse = 0.041
samples = 99.9%
value = 1.021
node #2
mse = 30.75
samples = 0.1%
value = 14.5
node #3
REVENUE ≤ 2458.6
mse = 0.0
samples = 98.4%
value = 1.0
node #4
REVENUE ≤ 7288.1
mse = 0.797
samples = 1.5%
value = 2.395
node #9
mse = 0.0
samples = 98.3%
value = 1.0
node #10
mse = 0.25
samples = 0.1%
value = 1.5
node #5
REVENUE ≤ 4137.5
mse = 0.112
samples = 1.4%
value = 2.128
node #6
mse = 0.0
samples = 0.1%
value = 5.0
node #7
mse = 0.0
samples = 1.1%
value = 2.0
node #8
REVENUE ≤ 4897.34
mse = 0.139
samples = 0.2%
value = 2.833
node #11
mse = 0.222
samples = 0.1%
value = 2.667
node #12
mse = 0.0
samples = 0.1%
value = 3.0

P7

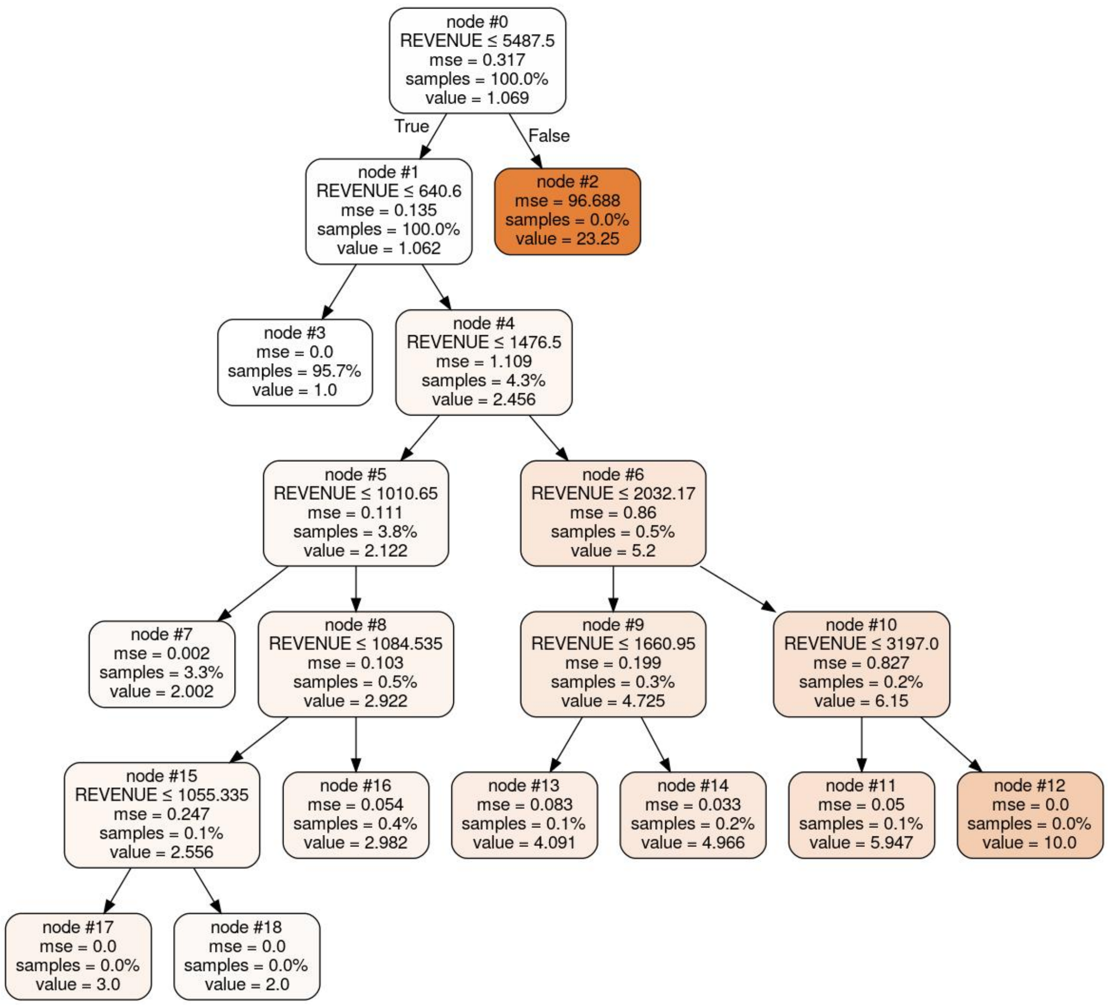


node #0
REVENUE ≤ 202044.25
mse = 2.086
samples = 100.0%
value = 1.081
True
False
node #1
REVENUE ≤ 1802.315
mse = 0.271
samples = 100.0%
value = 1.077
node #2
mse = 0.0
samples = 0.0%
value = 500.0
node #3
REVENUE ≤ 1069.59
mse = 0.036
samples = 98.9%
value = 1.038
node #4
REVENUE ≤ 12053.3
mse = 8.652
samples = 1.1%
value = 4.536
node #7
mse = 0.0
samples = 95.1%
value = 1.0
node #8
mse = 0.001
samples = 3.7%
value = 2.001
node #5
REVENUE ≤ 5969.95
mse = 2.435
samples = 1.1%
value = 4.314
node #6
REVENUE ≤ 29587.5
mse = 95.122
samples = 0.0%
value = 28.857
node #9
REVENUE ≤ 2712.575
mse = 1.019
samples = 1.1%
value = 4.091
node #10
mse = 2.212
samples = 0.0%
value = 10.463
node #13
REVENUE ≤ 15528.625
mse = 24.056
samples = 0.0%
value = 25.333
node #14
mse = 0.0
samples = 0.0%
value = 50.0
node #11
mse = 0.041
samples = 0.5%
value = 3.042
node #12
REVENUE ≤ 3258.125
mse = 0.307
samples = 0.6%
value = 4.879
node #15
mse = 3.0
samples = 0.0%
value = 19.0
node #16
mse = 4.5
samples = 0.0%
value = 28.5
node #17
mse = 0.159
samples = 0.2%
value = 4.198
node #18
mse = 0.157
samples = 0.5%
value = 5.098

P8


node #0
REVENUE ≤ 623.55
mse = 0.136
samples = 100.0%
value = 1.066
True
False
node #1
mse = 0.0
samples = 95.1%
value = 1.0
node #2
REVENUE ≤ 1506.5
mse = 1.054
samples = 4.9%
value = 2.343
node #3
REVENUE ≤ 1010.65
mse = 0.09
samples = 4.5%
value = 2.092
node #4
REVENUE ≤ 3052.5
mse = 2.927
samples = 0.4%
value = 5.293
node #7
mse = 0.001
samples = 4.1%
value = 2.001
node #8
REVENUE ≤ 1079.175
mse = 0.107
samples = 0.4%
value = 2.964
node #5
REVENUE ≤ 2020.8
mse = 0.56
samples = 0.4%
value = 4.944
node #6
mse = 4.25
samples = 0.0%
value = 11.5
node #13
REVENUE ≤ 1077.75
mse = 0.243
samples = 0.1%
value = 2.583
node #14
REVENUE ≤ 1355.85
mse = 0.056
samples = 0.4%
value = 3.028
node #9
REVENUE ≤ 1801.25
mse = 0.281
samples = 0.3%
value = 4.588
node #10
mse = 0.128
samples = 0.1%
value = 5.85
node #15
mse = 0.173
samples = 0.0%
value = 2.778
node #16
mse = 0.0
samples = 0.0%
value = 2.0
node #17
mse = 0.031
samples = 0.3%
value = 3.0
node #18
mse = 0.222
samples = 0.0%
value = 3.333
node #11
mse = 0.262
samples = 0.1%
value = 4.24
node #12
mse = 0.071
samples = 0.1%
value = 4.923

P9



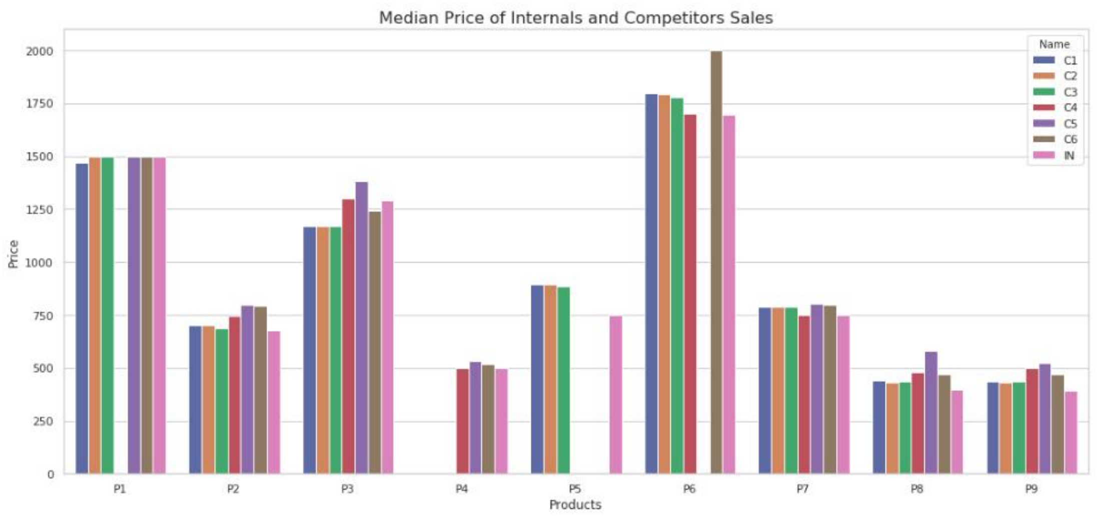
Conclusion

In addition to the model used, a comparison was made with the linear regression model, although the RMSE value is closer to 0, this model presents a larger amount of overfitting than the decision tree.

After the comparison between these two models, the GridSearchCV was applied to optimize the parameters of the models. Even after optimization the evaluation of the decision tree model continues to perform better than the others.

8. Competitor Analysis

Price Analysis

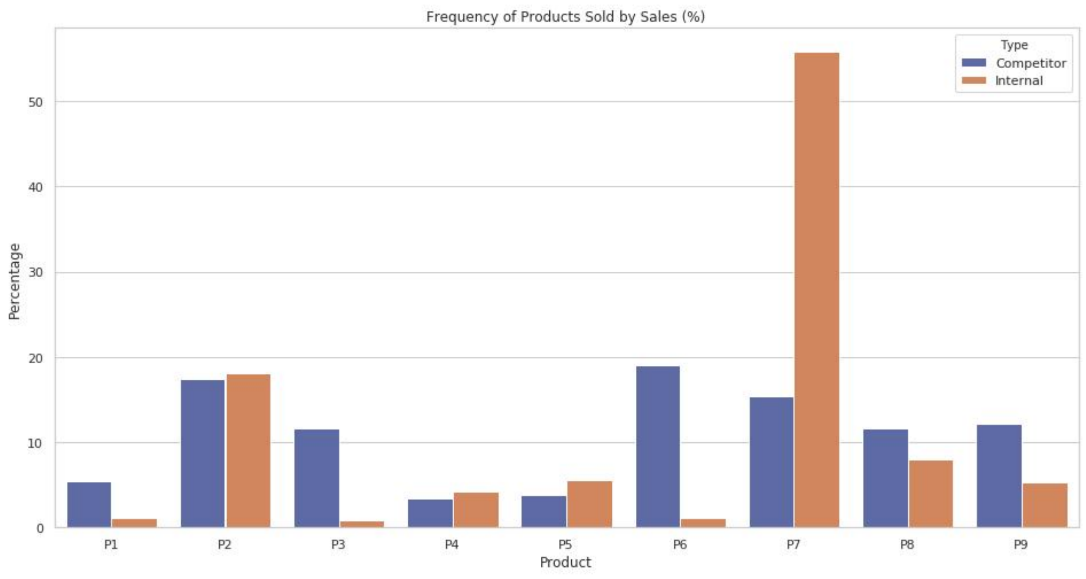


In the chart above, we can see the median sales prices for each product. The bar in pink, labeled "IN", represents the value of domestic sales.

Since competitors' sales do not have the quantity sold, the median value was selected as the parameter for this analysis, since domestic sales have many price variations because of the quantity sold.

We can observe that the median price of domestic sales has a value below compared to competitors.

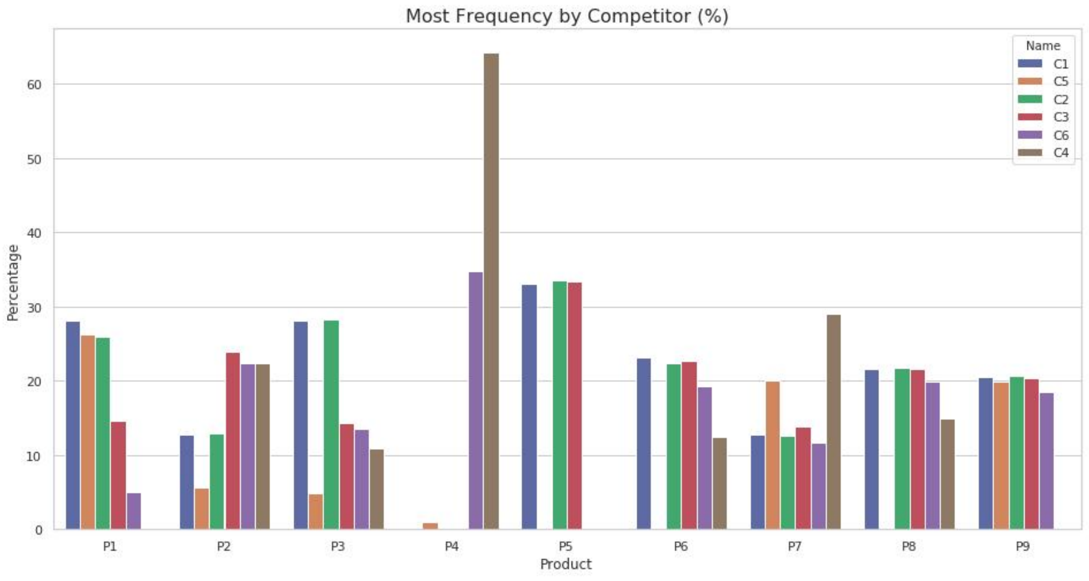
Frequency Analysis



The chart above shows the percentage of the frequency of products sold per sale performed, it counts each product contained in a sale and at the end divides over the total quantity, resulting in the above measure.

Analyzing the result of the graph, we can see that customers who make an internal purchase on the site, usually buy more products P2, P4, P5 and P7.

For the analysis of competitors, customers usually buy more products P1, P3, P5, P6, P8 and P9.



By drilldown on the products sold by competitors, we can analyze which of them sell the most.

In product P1 it can be observed that competitors C1, C2 and C2 have a higher frequency in sales compared to other competitors.

In product P3 the graph shows that competitors C1 and C2 have a higher frequency in sales compared to other competitors.

In product P5, we see a much higher frequency in sales for competitor C4 compared to other competitors.

In product P6, it can be observed that competitors C1, C2 and C3 have a higher frequency compared to other competitors.

In product P8, it can be observed that competitors C1, C2, C3 and C6 have a higher frequency compared to other competitors.

In the product P9, it is possible to observe the frequency for this product is balanced between competitors.

Conclusion

Looking at the analyzes made above, we can follow the behavior of competitors' sales for certain products. Even if the total value of revenue in domestic sales is higher, it is important to understand customer buying behavior for products sold by other players. If we can understand and convert these external sales to domestic sales, we can increase our revenue more and more.