

CornerRadius

by Uber



Desafio CornerShop

Jaime Arroyo León - Data Scientist

Expected Outcome:

Predict the `total time` an order placed through the **CornerRadius** app will take to optimize the user experience based on clear and real information delivered through the app according to this indicator.

Model Development:

The entire analysis was developed in `Visual Code Studio` for dissemination and analysis. The code for this challenge was shared via `github` (Restricted access for users with permissions). #Place github link

The code:

The following code installs the necessary libraries for model development. Due to the time for do this challenge, `pycaret` will be used for the model comparison according to R2 and MAE metrics. `Pycaret` is an open-source "low-code" library that allow training multiple models efficiently. This is commonly used for creating "proof of concept models". For more information about it, the following link is attached: <https://pycaret.org/>

For replicability purposes, "requirements.txt" is attached from python enviroment created for the development of this model. At the same time, you will find a file called

"functions.py" which contains all the functions used in this model.

<https://s3-us-west-2.amazonaws.com/secure.notion-static.com/72a6877b-c27f-4621-a33c-13175c37641d/requirements.txt>

```
#Importar librerias necesarias para el desarrollo del desafio
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
from pycaret.regression import *
from sklearn.model_selection import train_test_split
import gmaps
import os
import random
from geopy.distance import geodesic
from sklearn.linear_model import LinearRegression
import statsmodels.api as sm
from datetime import *

#Lectura de funciones pipeline Cornershop
import cornershop_functions.functions as corner_fx

#Magic commands
%matplotlib inline
!jupyter nbextension enable --py --sys-prefix widgetsnbextension
!jupyter nbextension enable --py --sys-prefix gmaps
```

For model development there are 4 .csv files which contain information related to orders executed in the **Cornershop** app. Once the data is loaded, use is made of the different functions integrated in the algorithm that allow the following to be discussed:

a) Training and testing dataset separation is performed according to indications provided in Readme. For the training dataset (which will be separated later in the training), 8,000 rows are kept leaving 2,000 rows for testing (deliverable).

```
#Lectura de base de ordenes
orders = corner_fx.read_dataset('./data/orders.csv')
print('[INFO]... Mostrando las 5 primeras filas del dataset 🤖')
orders.head()
```

[INFO]... Leyendo csv con separador ";" 📄
[INFO]... Leyendo csv con otro separador 📄
[INFO]... Mostrando las 5 primeras filas del dataset 🤖

	order_id	lat	lng	promised_time	on_demand	shopper_id	store_branch_id	total_minutes
0	e750294655c2c7c34d83cc3181c09de4	-33.501675	-70.579369	2019-10-18 20:48:00+00:00	True	e63bc83a1a952fa2b3cc9d558fb943cf	65ded5353c5ee48d0b7d48c591b8f430	67.684264
1	6581174846221cb6c467348e87f57641	-33.440584	-70.556283	2019-10-19 01:00:00+00:00	False	195f9e9d84a4ba9033c4b6a756334d8b	45fbc6d3e05ebd93369ce542e8f2322d	57.060632
2	3a226ea48debc0a7ae9950d5540f2f34	-32.987022	-71.544842	2019-10-19 14:54:00+00:00	True	a5b9ddcd0d82e61582fca19ad43dbaacb	07563a3fe3bbe7e3ba84431ad9d055af	NaN
3	7d2ed03fe4966083e74b12694b1669d8	-33.328075	-70.512659	2019-10-18 21:47:00+00:00	True	d0b3f6bf7e249e5ebb8d3129341773a2	f1748d6b0fd9d439f71450117eba2725	52.067742
4	b4b2682d77118155fe4716300cd7f39	-33.403239	-70.564020	2019-10-19 20:00:00+00:00	False	5c5199ce027b77caa9c2590a39ad27d	1f0e3dad99908345f7439f8fabbdfc4	140.724822

b) Using `"corner_fx.estadisticos"` you get basic data of each dataset and using `"corner_fx.unique_values"` you get descriptive information in relation to frequency and frequency accumulated by unique values.

order_products:

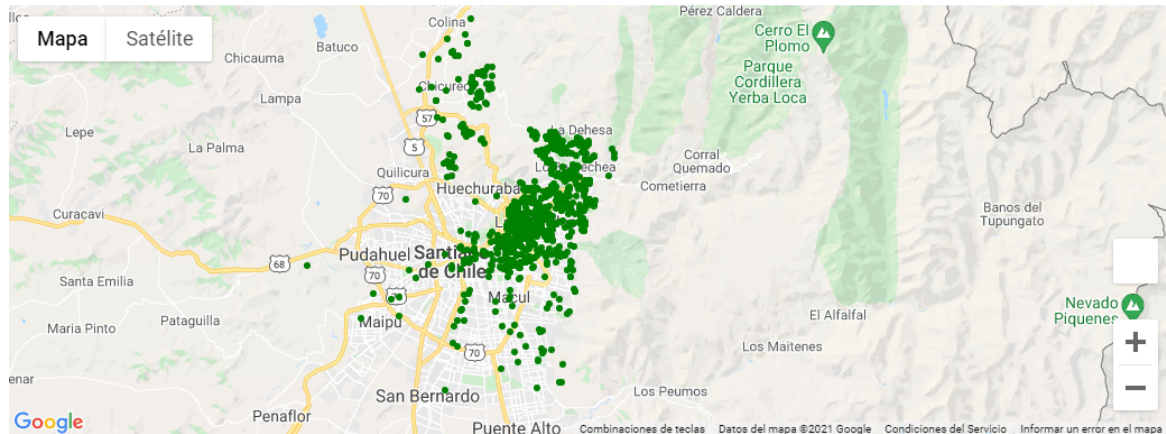
- 1) 198.500 rows and 4 columns.
- 2) None null values
- 3) Data on the quantity of products requested is retrieved for each order in **Units** and **Kg**. For products requested by "units", people add **18 products** in average and an average **2 Kg** in products labeled "**Kg**".
- 4) The 20 products most requested by customers are which represent 9.3% of the data. It is the table that show that.

Mostrando el top 20 categorias según frecuencia			
	Cantidad	Frecuencia	Frecuencia Acumulada
f90b55ea82b8af664f1d7eebb93f25e0	2073	1.044332	1.044332
bdcc4590896a0d0b419d0388fd16a859	1598	0.805038	1.849370
41da609c519d77b29be442f8c1105647	1562	0.786902	2.636272
280cf18baf4311c92aa5a042336587d3	1336	0.673048	3.309320
fb3336fca8851437f980e83045e75749	1092	0.550126	3.859446
42241c8a618f2d925c611ea9faaf44c4	876	0.441310	4.300756
f387d33f0f968f1005e9ef45b66266c6	836	0.421159	4.721914
deeb1b996407d124ff4ddf2f72a96ea0	818	0.412091	5.134005
03c3dafd39fe1c6abeb4bd72e3255d29	799	0.402519	5.536524
7e133b5ec7ad3911f79f03ebf0672200	790	0.397985	5.934509
2b80f98f17cdda9100f8b135f99a090d	755	0.380353	6.314861
d941f7183f6331a2c7d59cf1333e93c6	748	0.376826	6.691688
652cf38361a209088302ba2b8b7f51e0	737	0.371285	7.062972
9559fc73b13fa721a816958488a5b449	732	0.368766	7.431738
9c4a863a146054ac019d6f9167f2354a	704	0.354660	7.786398
1ddf2f2bddd3e706d89fec508c58a69c	699	0.352141	8.138539
1fba6e9254d8a0ae8706eb2560545949	658	0.331486	8.470025
7a123298c12ed5fcef8b17f6798535f	570	0.287154	8.757179
d9ccf24afa3ef38b3ca9388b6f296682	546	0.275063	9.032242
397c6acfeb22f9e9a1a8764db40d8808	544	0.274055	9.306297

orders:

- 1) 10.000 rows and 8 columns.

- 2) No null data. (Only label per construction of the challenge "total_minutes")
- 3) Using `google maps API` it is possible to verify that there are orders made in Region Metropolitana, La Serena, Viña del Mar and Concepción.
- 4) In relation to Region Metropolitana, it can be seen that the orders delivered are concentrated in the central-eastern sector of the capital.



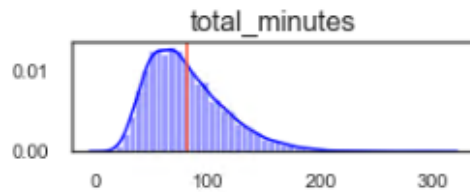
```
[INFO] Frecuencia por dato unico para campo comuna 🚀 ...
```

Cantidad de valores unicos por categoría: 115

Mostrando el top 10 categorías según frecuencia

	Cantidad	Frecuencia	Frecuencia Acumulada
Las Condes	2066	25.8250	25.8250
Lo Barnechea	1254	15.6750	41.5000
Vitacura	1042	13.0250	54.5250
Providencia	532	6.6500	61.1750
Ñuñoa	357	4.4625	65.6375
Barrio El Golf	354	4.4250	70.0625
La Reina	284	3.5500	73.6125
Chicureo	279	3.4875	77.1000
Viña del Mar	214	2.6750	79.7750
Santiago	195	2.4375	82.2125

- 5) Orders have an average of 81 minutes. With some edge cases whose total time exceeds 150 minutes.



6) 54.5% of the orders were placed during hours of low demand.

```
[INFO]
Frecuencia por dato unico para campo on_demand 🚀 ...
```

	Cantidad	Frecuencia	Frecuencia Acumulada
False	4362	54.525	54.525
True	3638	45.475	100.000

7) The two most recurring stores are 1f0e3dad99908345f7439f8ffabdfcc4 and 1679091c5a880faf6fb5e6087eb1b2dc which represent 35.28% of the orders in the data.

shoopers:

- 1) 2.864 rows and 6 columns.
- 2) The next variables have null values.

```
[INFO]... Cantidad de datos nulos por columna: 🚫
```

shopper_id	0
seniority	0
found_rate	101
picking_speed	0
accepted_rate	27
rating	84
dtype:	int64

- 3) Shoppers on average find 86% of products with an average speed of 2 products per min. It is a base of shoppers with ratings very close to 5, whose 25% percentile is 4.8 and an average of 4.85 points. The historical rate for each shopper is 90%.
- 4) Given the previous analysis, it is possible to infer that "**picking_speed**" is one of the variables that could be of importance for the prediction. The rest are homogeneously distributed without significant differences.

```
[INFO]... Estadisticos basicos: 🔄
```

	found_rate	picking_speed	accepted_rate	rating
count	2763.000000	2864.000000	2837.000000	2780.000000
mean	0.861082	1.762392	0.908276	4.848428
std	0.031038	0.665962	0.107911	0.133011
min	0.737300	0.650000	0.240000	3.880000
25%	0.842900	1.290000	0.880000	4.800000
50%	0.863900	1.580000	0.944444	4.880000
75%	0.881950	2.120000	1.000000	4.960000
max	0.971000	7.040000	1.000000	5.000000

5) To fill na the missing data, the "seniority" field and the averages related to each metric are used. That is, averages are generated for each "seniority" group and each one of them is imputed in the missing values. This fill can be improved by using interaction of variables related to similar profiles of shoppers.

```
[INFO]... Completando datos vacios en shoppers 🔄
[INFO]... Completando datos vacios columna found_rate_fillna: 🔄
[INFO]... Completando datos vacios columna picking_speed_fillna: 🔄
[INFO]... Completando datos vacios columna accepted_rate_fillna: 🔄
[INFO]... Completando datos vacios columna rating_fillna: 🔄
[INFO]... Cantidad nulos: 🚫
shopper_id      0
seniority        0
found_rate       0
picking_speed    0
accepted_rate    0
rating           0
dtype: int64
[INFO]... Mostrando los 5 primeros registros de la base post tratamiento 📄
```

	shopper_id	seniority	found_rate	picking_speed	accepted_rate	rating
0	1fc20b0bdf697ac13dd6a15cbd2fe60a	41dc7c9e385c4d2b6c1f7836973951bf	0.860600	1.94	1.00000	4.870000
1	e1c679ac73a69c01981fdd3c5ab8beda	6c90661e6d2c7579f5ce337c3391dbb9	0.844600	1.23	0.92000	4.920000
2	09d369c66ca86ebffac133410c5ee1	6c90661e6d2c7579f5ce337c3391dbb9	0.855900	1.56	1.00000	4.880000
3	db39866e62b95bb04ebb1e470f2d1347	50e13ee63f086c2fe84229348bc91b5b	0.846954	2.41	0.78644	4.843834
4	8efbc238660053b19f00ca431144fdae	6c90661e6d2c7579f5ce337c3391dbb9	0.877000	1.31	0.92000	4.880000

6) It has 4 groups of "seniority" which the first one represent 57.36% of the data.

[INFO]

Frecuencia por dato unico para campo seniority 🚀 ...

	Cantidad	Frecuencia	Frecuencia Acumulada
6c90661e6d2c7579f5ce337c3391dbb9	1643	57.367318	57.367318
50e13ee63f086c2fe84229348bc91b5b	719	25.104749	82.472067
41dc7c9e385c4d2b6c1f7836973951bf	440	15.363128	97.835196
bb29b8d0d196b5db5a5350e5e3ae2b1f	62	2.164804	100.000000

storebranch:

- 1) 476 rows and 4 columns.
- 2) No null data.
- 3) Latitude and longitude of the different stores are retrieved for the purposes of generating new variables.

Creating Variables:

- a) The `geodesic` library is used to measure the distance between the place where the order was requested and the assigned store. With this, a "distance" field is created which gives the Km that exist between both places.
- b) The day, month, year, hour and minute of the order are obtained. For analysis and development purposes, "promised_time" is used since the original date of the order does not appear in the dataset. (With this field it could be more exact, avoiding biases and generating optimization analysis for the predictions of each model).

```
[INFO]
Frecuencia por dato unico para campo dia_pedido 🚀 ...
      Cantidad  Frecuencia  Frecuencia Acumulada
19      5673      70.9125      70.9125
18      2326      29.0750      99.9875
20         1       0.0125     100.0000

[INFO]
Frecuencia por dato unico para campo mes_pedido 🚀 ...
      Cantidad  Frecuencia  Frecuencia Acumulada
10      8000      100.0      100.0

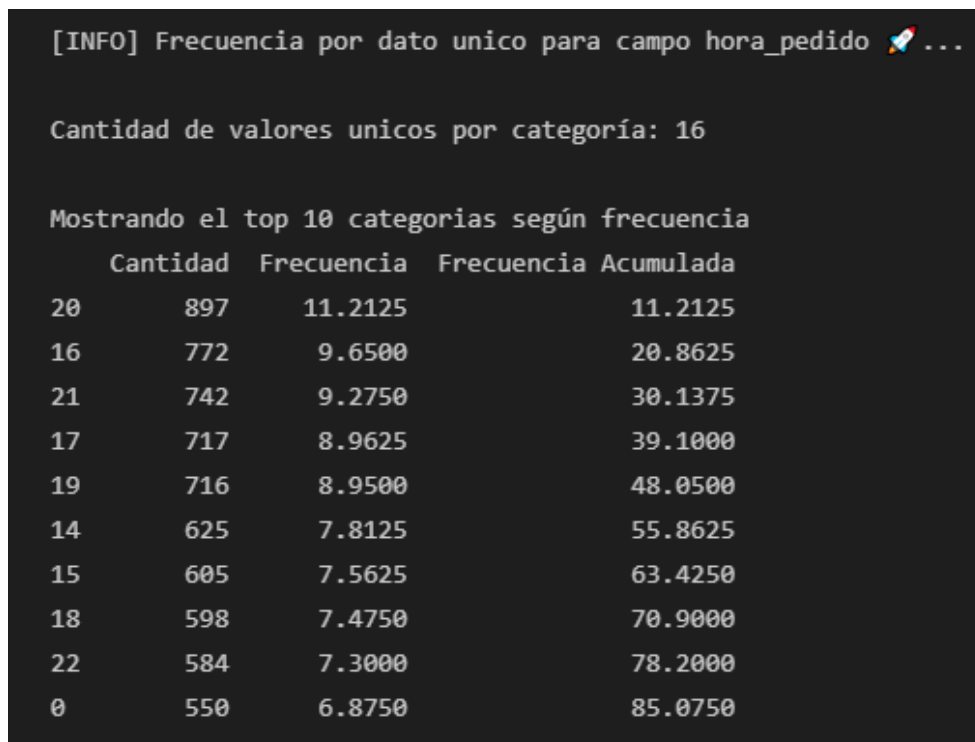
[INFO]
Frecuencia por dato unico para campo año_pedido 🚀 ...
      Cantidad  Frecuencia  Frecuencia Acumulada
2019      8000      100.0      100.0
```

c) In relation to the previous variables, the dataset only incorporates two days of orders, for October 2019.

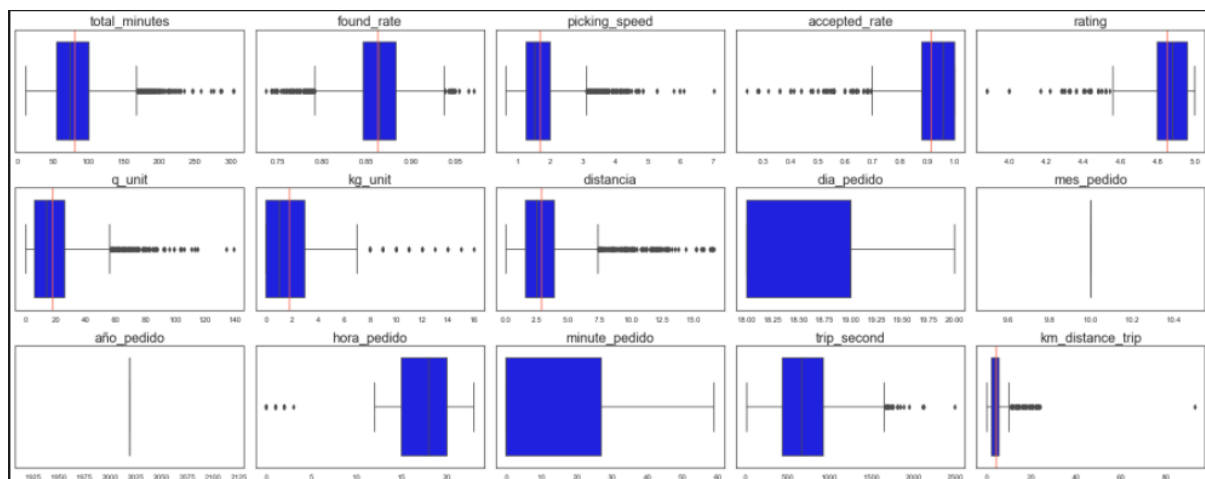
d) By using `Google's API Distance Matrix`, the distance the shopper must travel and the minutes it takes to get from point A to B. Due to the API only allow to get the real time traffic (In present or future), the data was used with updated date and time for investigation purpose. In the case of mounting the model online and that the variable is significant, it can be obtained by consuming the API. To avoid consuming the API, dataset `maps_results.csv` was made available.

e) Through the use of `Google Geocoding API`, "la comuna" from where the order is requested was obtained. This API provides another type of information which for simplicity purposes only this data was obtained. To avoid consuming the API, a dataset `comunas.csv` was made available.

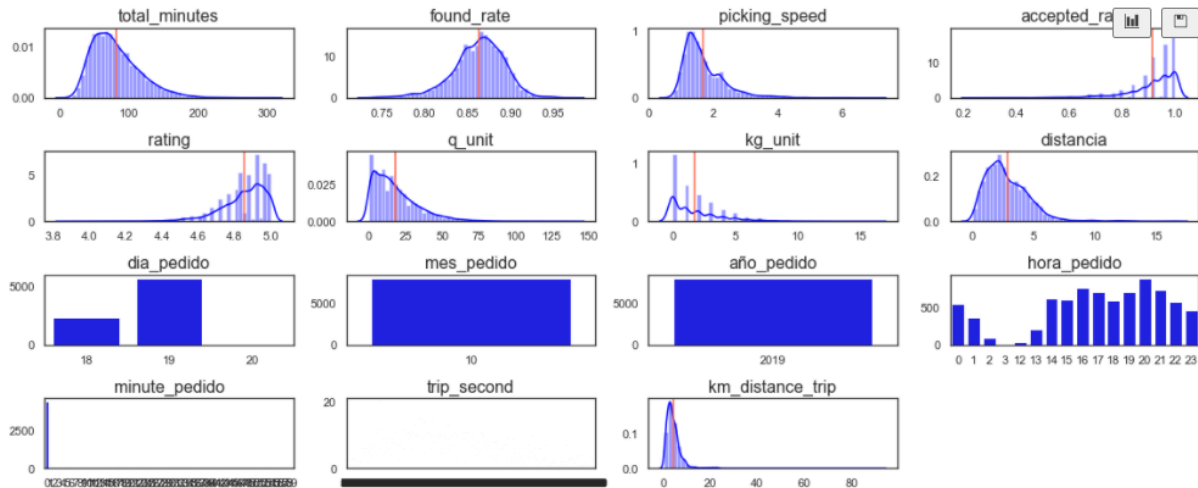
f) It is possible to identify that more than 50% of the orders occur after 4:00 p.m.



g) Once we have the complete dataset, we proceed to apply the `"plot_boxplot"` function to check if the variables have **outliers**. Here it is possible to verify that the delivered data contains a high level of outliers. Considering that there are very few samples to train the model, it is probable that the results in the prediction will not be able to generalize correctly. (Even more so when outliers were removed in training)

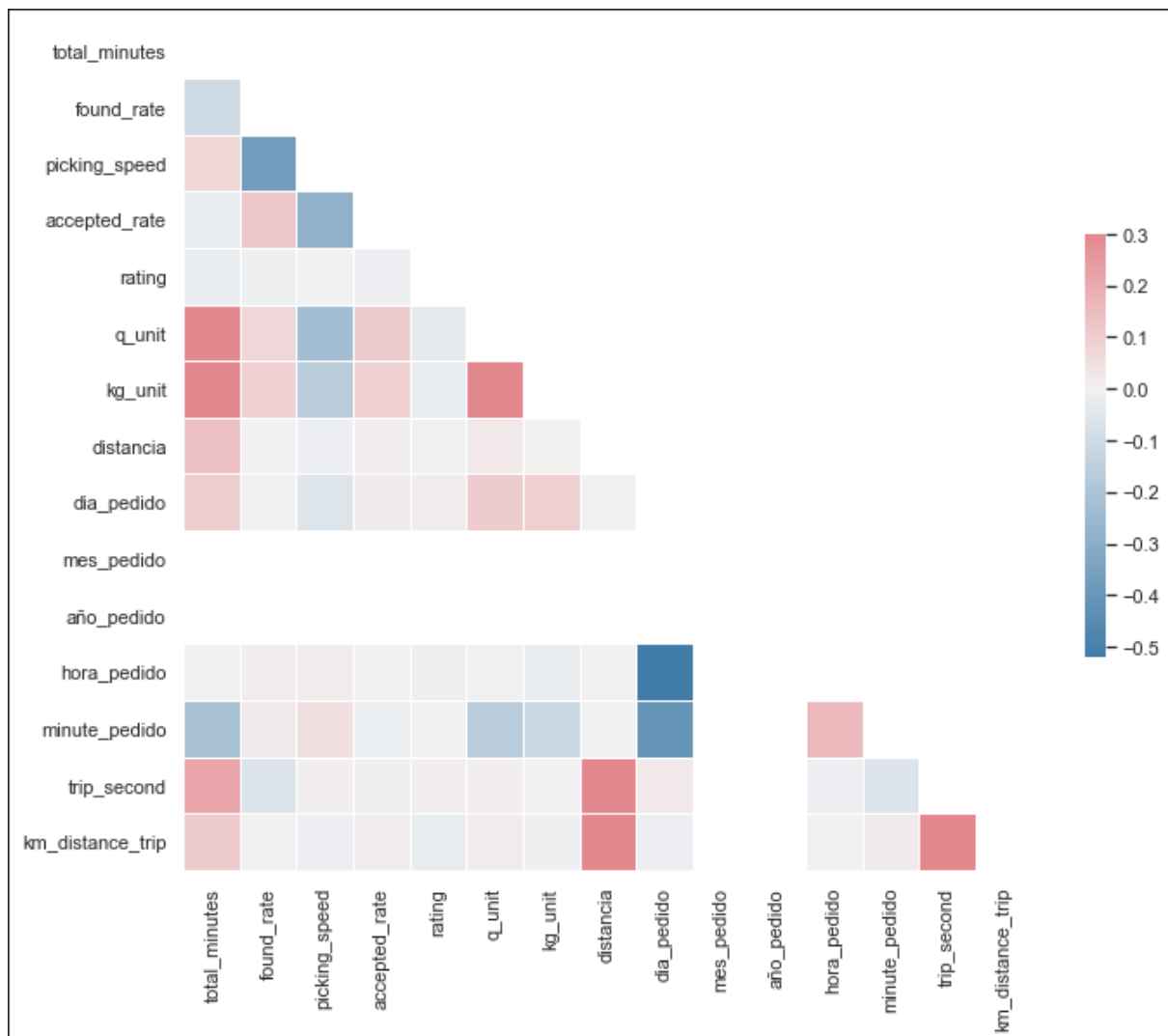


h) With function `"plot_hist_freq"` the distributions are obtained over the total of variables generated. Here it is possible to show variables with data distributed outside the norm.



i) Before training the model, apply `"LabelEncoder"` from sklearn through the function `"cat_encoding"` to convert categorical variables into numeric, keeping their type for training different models.

j) By calculating the correlation between the variables it is possible to see that **found_rate, q_unit, kg_unit, trip_second and distancia** are variables correlated with the target variable. In the training process, variables with correlation will be eliminated, maintaining those of greater significance for training.



Entrenamiento del modelo:

The following line of code, configures the necessary parameters for the application of `Pycaret` to our dataset. In this configuration, it is indicated which is the data to be used, which will be the target to be predicted, applying standardization to the variables and specifying which are categorical and which are numerical. For training purposes, `Pycaret` eliminates `outliers` (using inter-quartile ranges), eliminates correlated variables and uses Bayesian optimization of `hyperparameters` to obtain the best `MAE`.

```
#Aplicacion de Pycaret
model = setup(df_out,
              target='total_minutes',
              normalize = True,
              normalize_method = 'zscore',
              train_size = 0.85,
```

```

categorical_features = df_out.select_dtypes(include = [np.object]).columns.tolist(),
numeric_features = int_list,
#remove_outliers = True,
#outliers_threshold = 0.05,
remove_multicollinearity = True,
multicollinearity_threshold = 0.9,
fold_shuffle = True,
feature_selection = True,
feature_selection_threshold = 0.8,
session_id = 1992)

# Comparando modelos
compare_models()

```

`Compare_models()` allows training multiple ML models using Cross-Validation with 10 folds, separating the data into 85% training and 15% for validation. (This configuration was chosen since there is very little data to train).

The result of the process is as follows:

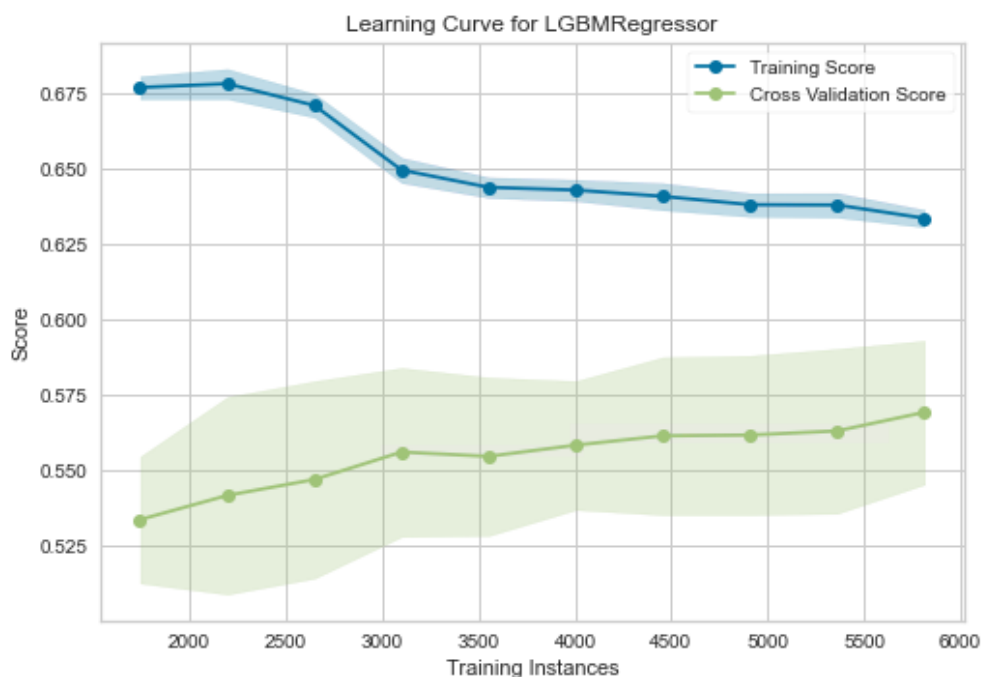
	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
catboost	CatBoost Regressor	16.4557	484.5624	21.9976	0.6000	0.2658	0.2243	15.2570
lightgbm	Light Gradient Boosting Machine	16.6051	496.6183	22.2690	0.5902	0.2661	0.2246	1.6650
xgboost	Extreme Gradient Boosting	16.9003	512.2027	22.6159	0.5772	0.2716	0.2296	26.8610
rf	Random Forest Regressor	16.8968	529.6796	22.9987	0.5629	0.2722	0.2258	20.2980
br	Bayesian Ridge	17.2477	532.0925	23.0483	0.5605	0.2783	0.2351	74.2940
et	Extra Trees Regressor	16.8132	535.9047	23.1314	0.5580	0.2706	0.2179	32.2360
gbr	Gradient Boosting Regressor	17.3891	535.7651	23.1279	0.5580	0.2818	0.2413	15.3360
ridge	Ridge Regression	17.7667	562.3738	23.6952	0.5353	0.2967	0.2430	1.2730
omp	Orthogonal Matching Pursuit	18.2016	627.7894	25.0059	0.4812	0.3051	0.2465	0.9640
lasso	Lasso Regression	19.5080	663.5258	25.7360	0.4525	0.3143	0.2746	1.0190
knn	K Neighbors Regressor	19.7009	688.5257	26.2190	0.4318	0.3156	0.2690	9.3610
en	Elastic Net	20.5070	720.0037	26.8142	0.4059	0.3311	0.2974	0.5120
par	Passive Aggressive Regressor	20.6523	742.0281	27.2286	0.3864	0.3778	0.2814	3.0060
dt	Decision Tree Regressor	23.2209	989.6001	31.4416	0.1823	0.3756	0.3070	0.9310
ada	AdaBoost Regressor	27.6851	1052.6326	32.4302	0.1298	0.4347	0.4692	24.1190
huber	Huber Regressor	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	32.7550
llar	Lasso Least Angle Regression	27.3313	1213.9027	34.8306	-0.0019	0.4284	0.4087	0.9070
lr	Linear Regression	2013.1709	382187724.0500	14891.9941	-306375.3122	1.3067	28.3902	11.3090
lar	Least Angle Regression	1103.4113	8014952010.0035	28332.0593	-7043825.8934	0.3116	14.2010	2.0750

The models are automatically ordered from best to worst performance. However, due to metric purposes, the best model for this dataset is "**CatBoost Regressor**" with **MAE** and **R2 score** of **16.46** and **0.6** respectively. Comparing results in validation, an **R2 score** of **0.56** is obtained, which indicates that the model is not overfitting.

Looking for an improvement in the results, we proceed to apply the function "`tune_model`" of `pycaret`, which optimizes the **hyperparameters** of the chosen model (**catboost**) using

Bayesian optimization to avoid the **overfitting** of parameters. After applying, the results tend to improve. The following table shows the K-Fold 10 for the training dataset.

	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	16.2324	499.2895	22.3448	0.5612	0.2725	0.2297
1	15.6642	426.1623	20.6437	0.6341	0.2529	0.2146
2	16.9239	514.4272	22.6810	0.5888	0.2695	0.2292
3	16.6308	484.5198	22.0118	0.6257	0.2617	0.2261
4	15.9552	426.6907	20.6565	0.6345	0.2572	0.2176
5	16.6859	521.0662	22.8269	0.5907	0.2706	0.2252
6	16.8236	532.0528	23.0663	0.5914	0.2628	0.2153
7	16.4682	454.2844	21.3139	0.5984	0.2725	0.2339
8	16.6720	485.3699	22.0311	0.5945	0.2656	0.2241
9	16.5010	501.7610	22.4000	0.5809	0.2725	0.2275
Mean	16.4557	484.5624	21.9976	0.6000	0.2658	0.2243
SD	0.3760	35.5943	0.8173	0.0228	0.0066	0.0062



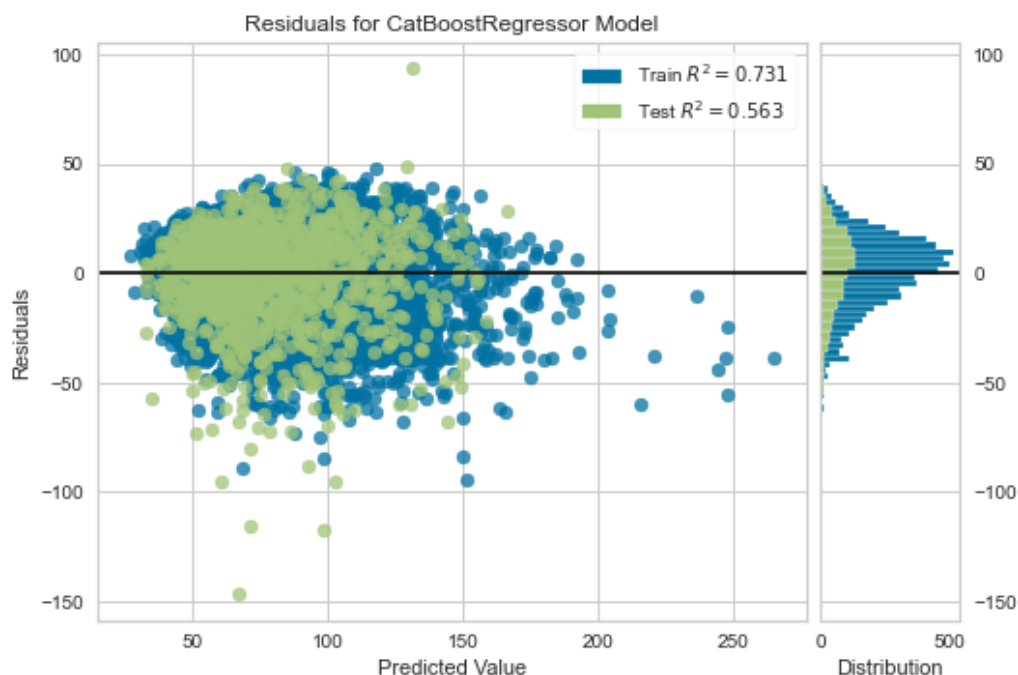
In order to better understand the output of the model, an optimization function should be used to obtain the actual prediction potential. Using the original date and time of the order, the minutes that the order actually took empirically and the estimated time delivered in the app to the client, it is possible to set up groups to measure the level of accuracy that the model achieves. For example, generating 3 groups: **1) In time:** real

delivery time vs estimated < 5 min, **2) Slight delay:** 15 min ≤ real delivery time vs estimated ≥ 5 min, **3) Delay:** Actual vs estimated delivery time > 15 min.

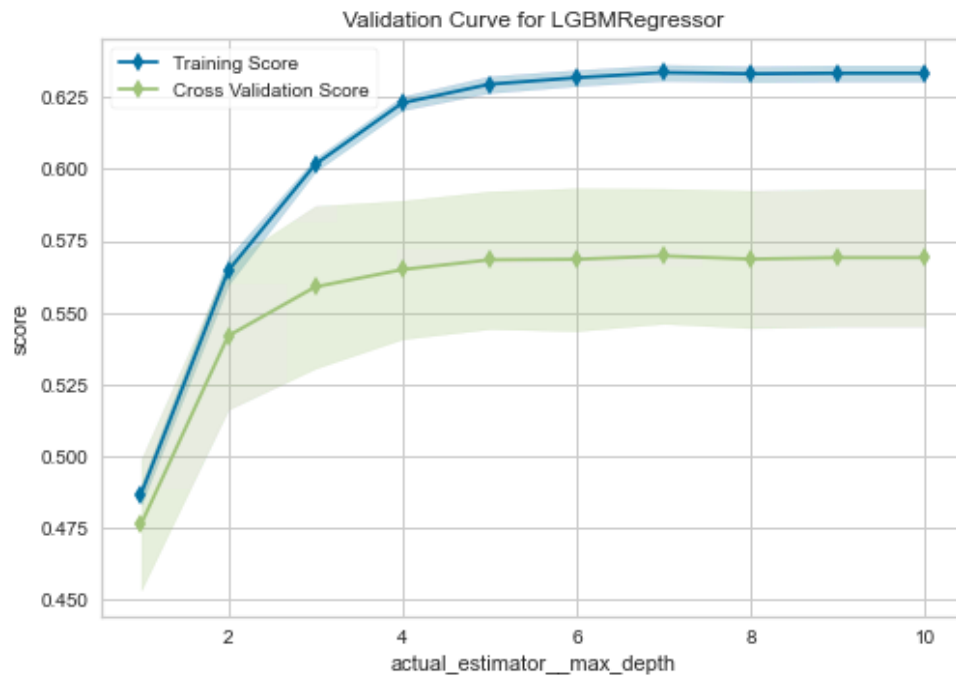
By obtaining the base distribution of these groups, with the new predictions obtained, it could be validated if our new model manages to optimize in time by increasing the **proportion** of cases "In time", significantly reducing the cases "Delay" and slightly the cases "Slight delay" in order to deliver a rich user experience based on the information provided in the application.

Importance of variables and other graphs (pycaret):

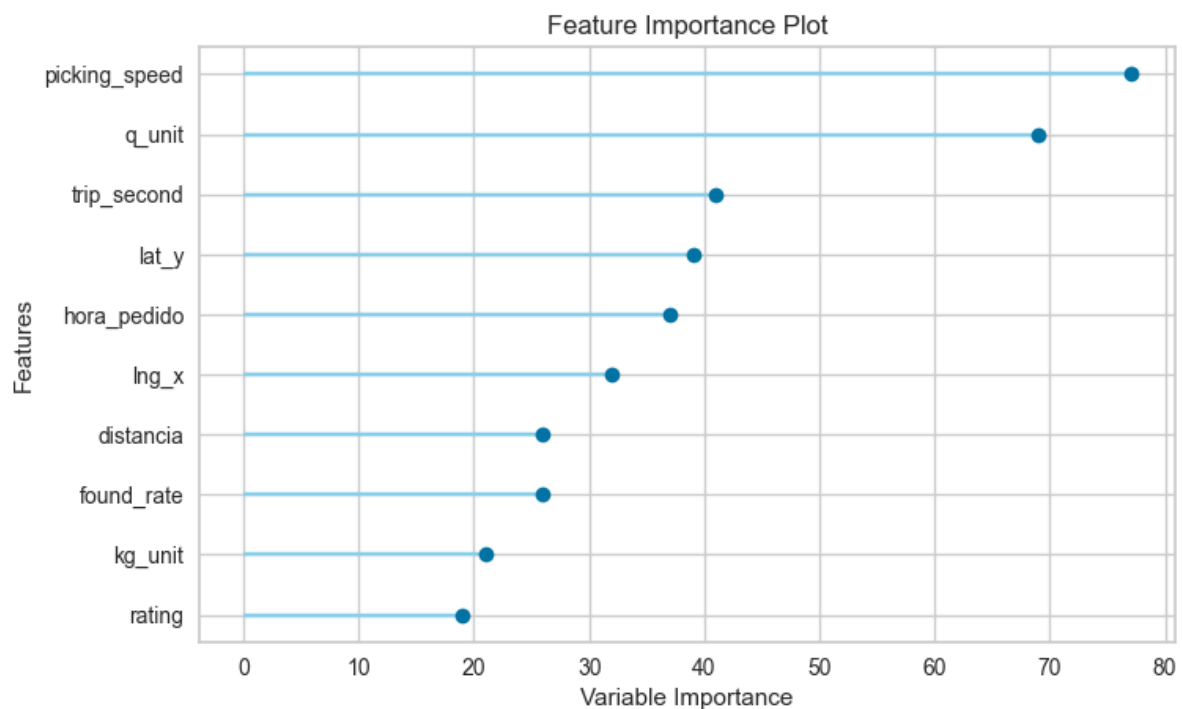
Residuals: The next graph shows how predictions are related to the real value giving their predictions residuals. The target is to closing all the points to 0. It has a normal distribution with mean close to 0 for training and testing.



Validation Curve: Allow to check how moving the hyperparameters it could get R2 score for each configuration. In this case, to optimize the metric, max_depth has to be 4.



Feature Importance: Plotting the most significance variables for the model. All of them make sense in specific the picking speed of the cshopper, the number of items that the customer add to the order and the distance or trip_second (driving).



Learning Curve: El siguiente grafico permite conocer la curva de aprendizaje del modelo utilizado, cuyo training y testing demuestran resultado muy similares los cuales permiten indicar que el modelo no se encuentre en overfitting.


Shap Value:

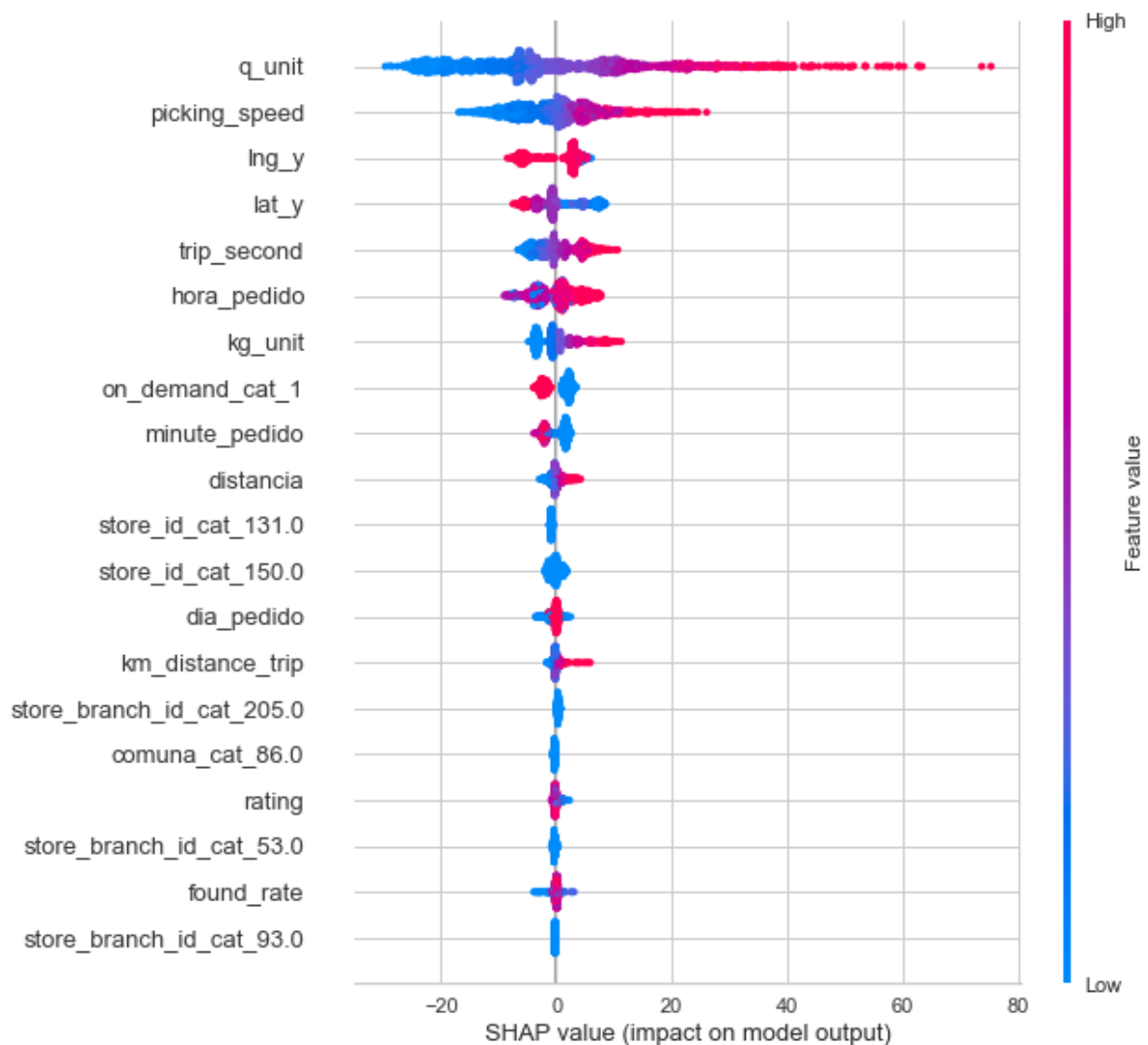
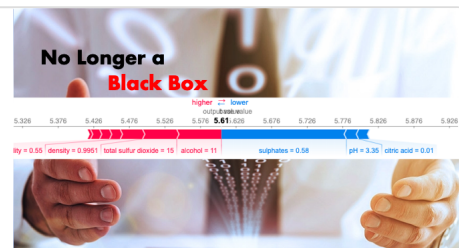
An interesting graph to observe is the attachment. **SHAP value** allows knowing the positive and negative impact on the interaction of each variable with the variable to be predicted. It is a graph that accompanies the importance of variables that allow us to better interpret the reason for the selection of each variable and its predictive power.

To know more about it, click the next link:

Explain Your Model with the SHAP Values

Better Interpretability Leads to Better Adoption Is your highly-trained model easy to understand? A sophisticated machine learning algorithms usually can produce accurate predictions, but

 <https://towardsdatascience.com/explain-your-model-with-the-shap-values-bc36aac4de3d>



Model Improvements:

Here are different ways to improve the model:

a) Increasing number of samples: By using more examples, you allow the model to acquire patterns and generalize better during training.

b) Adding New Variables: The variables of the delivered dataset seem not to be enough to improve the predictive power of our model. It is for this reason that it could be retrained with variables such as:

- 1) Brand and Model of the vehicle where the shopper deliver the order. (Motor's HP)
- 2) Type of vehicle (Moto, Car, etc)
- 3) Wheather conditions at the time of the order is entered.
- 4) Tip for the shopper that the customer put in the order.
- 5) If the customer lives in buildings or home.
- 6) More Geographic information

c) Use different methodology: In order to acquire behavioral data, one way to increase predictive power depending on the type of case study is to use `"time series"`. Given past events, estimate what will happen in the future.

d) Use feature engineering libraries: Make use of libraries such as `"featuretools"` in order to use `Deep Feature Synthesis` generating x amount of behavior variables that allow increasing the performance of the model.

e) Neural networks: Using `LSTM` recurrent neural networks could be a good way to compare current results and show whether the model improves when using `Deep Learning`.