INDICE

1.- Introducción

2.- Uso de BQML para predecir usuarios cobrados.

3.- Conclusiones

4.- Documentación

1.- INTRODUCCIÓN

Se van a intentar emplear técnicas de ML para la construcción de modelos de predicción que sean capaces de aumentar el rendimiento en el caso de negocio de DVS. Para esto vamos a usar la aplicación de Machine Learning de BigQuery, se van a explorar los datasets de Google Analytics en BigQuery, se optimizarán y evaluarán distintos modelos hasta conseguir uno aceptable con el cual haremos predicciones de resultados. Después y por último, compararemos estas predicciones con los datos reales y se extraerán conclusiones.

2.- USO DE BIGQUERY ML PARA PREDECIR USUARIOS COBRADOS

El objetivo de este ejercicio es predecir qué usuarios van a ser cobrados y cuáles no en un día concreto. El dataset empleado son los datos reales de Google Analytics de Digital Virgo Spain (DVS).

Como primer acercamiento a los datos, los visitantes del sitio web de DVS pueden suscribirse a sus servicios, lo que se contabilizará como transacción. Después DVS puede conseguir cobrarles la suscripción o no, dependiendo del estado de facturación de su línea telefónica. El % de conversión a transacción es menor a un 1%, y de la transacción al cobro (first billing) un 70% aproximadamente. El objetivo es predecir este first billing para invertir sólo en las transacciones que van a ser cobradas.

Para elaborar este ejercicio se usará la interfaz gráfica de BigQuery ML.

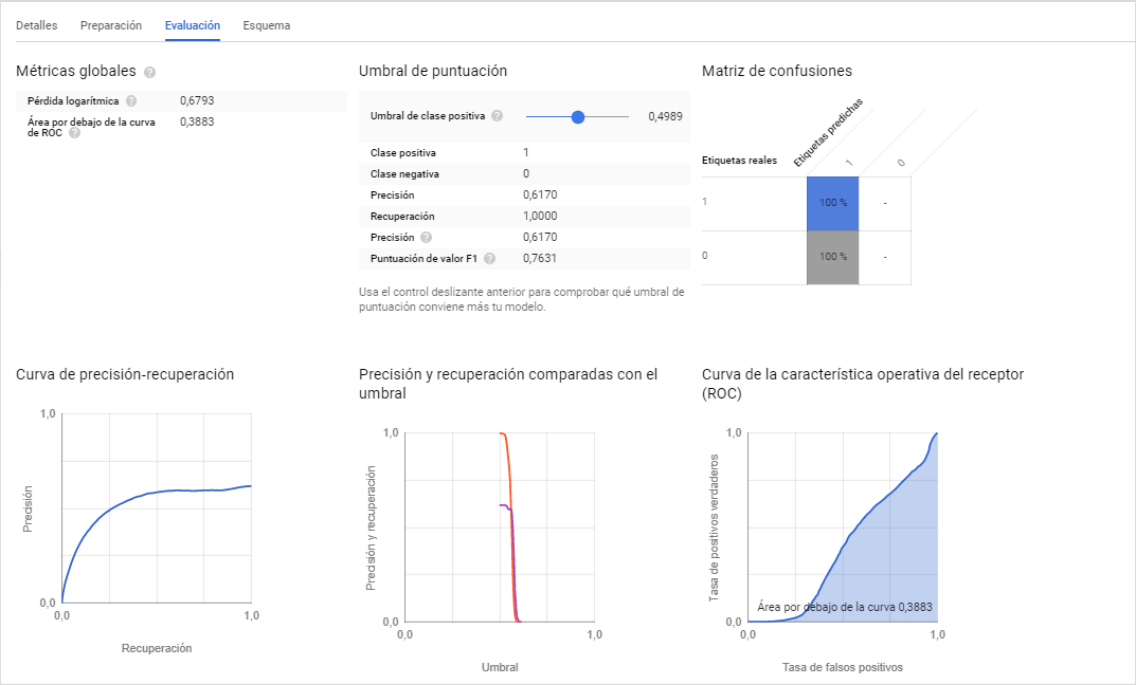
Modelo 1

Para elaborar el primer modelo, se introdujeron solamente los usuarios que tenían suscripción, además de variables como la fecha, la hora en formato “momento del día”, marca del dispositivo, navegador; y métricas como número de sesiones y transacciones. La consulta quedó así:

Se usó solo a los usuarios suscritos ya que con el ratio de conversión actual, el dataset es muy desbalanceado.

```
-- Modelo1
CREATE OR REPLACE MODEL `wide-oasis-135923.iamarketingdvs.modelodefinitivo1`
OPTIONS(model_type='logistic_reg') AS
SELECT
IFNULL(SUM(( SELECT value FROM UNNEST(hits.customMetrics) WHERE index = 2)),0) AS label,
date, case when hits.hour > 7 and hits.hour < 13 then 'manana' when hits.hour > 13 and
hits.hour < 20 then 'tarde' else 'noche' end as momentodia, hits.hour as hora, clientId,
geoNetwork.city as ciudad, device.deviceCategory as categoriaDispositivo,
device.mobileDeviceBranding as marca, device.browser as navegador, trafficSource.source
as fuente, trafficSource.campaign as campana, (SELECT value FROM
UNNEST(session.customDimensions) WHERE index = 14 GROUP BY 1) AS producto,
device.operatingSystem as OS, sum(totals.visits) as visitas, CASE WHEN
sum(totals.transactions) < 1 THEN 0 ELSE totals.transactions END as subs
FROM `wide-oasis-135923.140393857.ga_sessions_*` as session,
UNNEST(hits) AS hits
where _table_suffix BETWEEN '20190101' AND '20191001'
and trafficSource.campaign like '%range%'
and trafficSource.source like '%google%'
and totals.transactions > 0
GROUP BY date, clientId, ciudad, marca, categoriaDispositivo, totals.transactions,
navegador, campana, fuente, producto, OS, hora
```

Los resultados de la evaluación fueron estos:



Como se puede ver, la evaluación del modelo es mala (y un tanto extraña), basándonos en el AUC < 0.5, realmente sería peor que lanzar una moneda. Esto es incongruente porque no hay nada más aleatorio que lanzar una moneda, por tanto un área inferior al 50% lo que nos indica es que el modelo está mal formado, hay fallos graves de diseño.

Modelo 2

Fijándome en las variables utilizadas, caigo en la cuenta de que algunas son innecesarias (como por ejemplo el nombre de campaña o la fuente de tráfico, casi siempre la misma) y otras siempre tienen el mismo valor, como las transacciones (siempre 1).

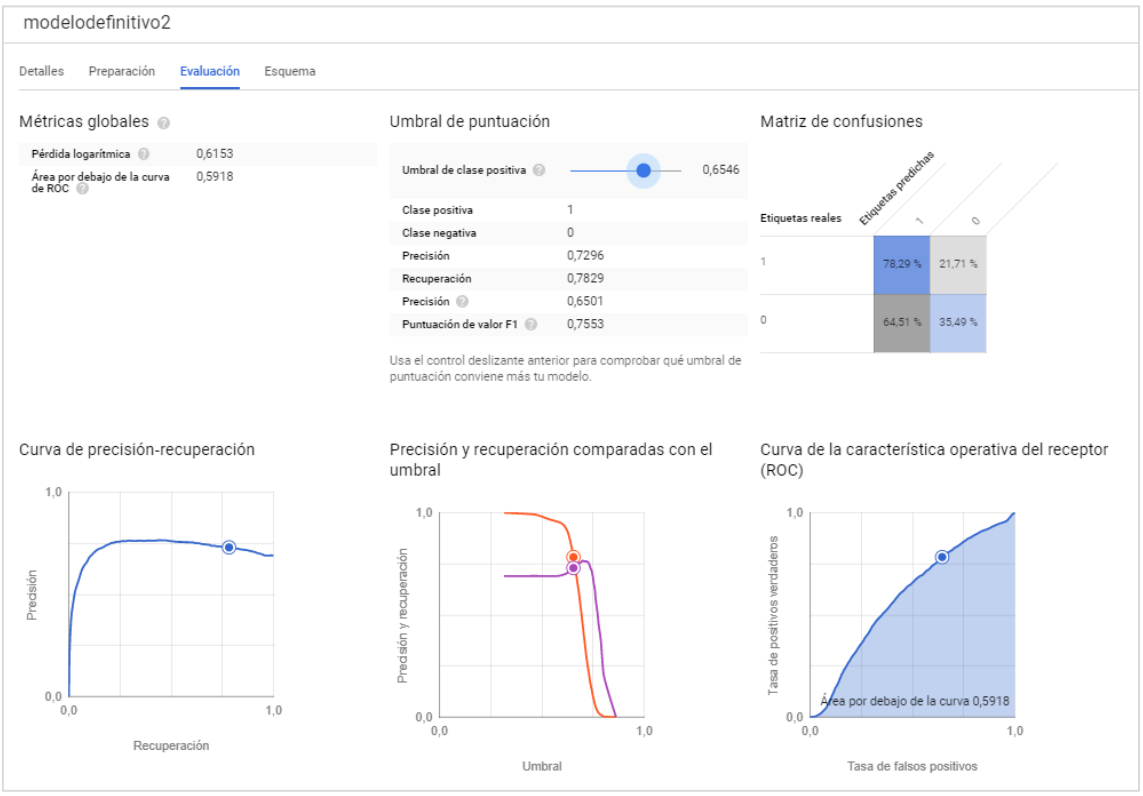
Elaboraremos entonces un segundo modelo ajustando los parámetros con los que componemos el dataset de train.

La consulta para la creación del segundo modelo quedó de esta forma:

```
CREATE OR REPLACE MODEL `wide-oasis-135923.iamarketingdvs.modelodefinitivo2`
OPTIONS(model_type='logistic_reg') AS
SELECT
IFNULL(SUM(( SELECT value FROM UNNEST(hits.customMetrics) WHERE index = 2)),0) AS label,
date, case when hits.hour > 7 and hits.hour < 13 then 'manana' when hits.hour > 13 and
hits.hour < 20 then 'tarde' else 'noche' end as momentodia, clientId, geoNetwork.city as
ciudad, device.mobileDeviceBranding as marca, device.browser as navegador,
trafficSource.campaign as campana, (SELECT value FROM UNNEST(session.customDimensions)
WHERE index = 14 GROUP BY 1) AS producto, device.operatingSystem as OS,
sum(totals.visits) as visitas
FROM `wide-oasis-135923.140393857.ga_sessions_*` as session,
UNNEST(hits) AS hits
where _table_suffix BETWEEN '20190101' AND '20191001'
and trafficSource.campaign like '%range%'
and trafficSource.source like '%google%'
```

```
and totals.transactions > 0
GROUP BY date, clientId, ciudad, marca, navegador, campana, producto, OS, momentodia
```

Su evaluación fue esta:



Primero de todo, vemos que un área bajo la curva > 0.5, esto quiere decir que ya tenemos algo mejor que lanzar una moneda al aire, por lo que sabemos positivamente que el ajuste al anterior modelo es positivo, había tanto dimensiones como métricas que no deben estar. Al obtener un modelo con área > 0.5, en la herramienta para evaluar modelos de BQ ML lo primero que vamos a hacer es elegir un umbral de clasificación conveniente para nuestro objetivo (predecir primer cobro en las transacciones). Usando el control deslizante y chequeando cómo va variando la matriz de confusión, decido que 0,6546 es el umbral de clasificación que mejor resultados nos va a reportar, ya que es el que más “distancia” obtiene entre los verdaderos positivos y los falsos positivos. 78,29% de verdadero positivo y 64,51% de falso positivo. Es cierto que hay muchos falsos positivos, pero no olvidemos que nuestro modelo tan solo tiene un 0.59 de área bajo la curva ROC, por tanto no parece que podemos aspirar a mucho más.

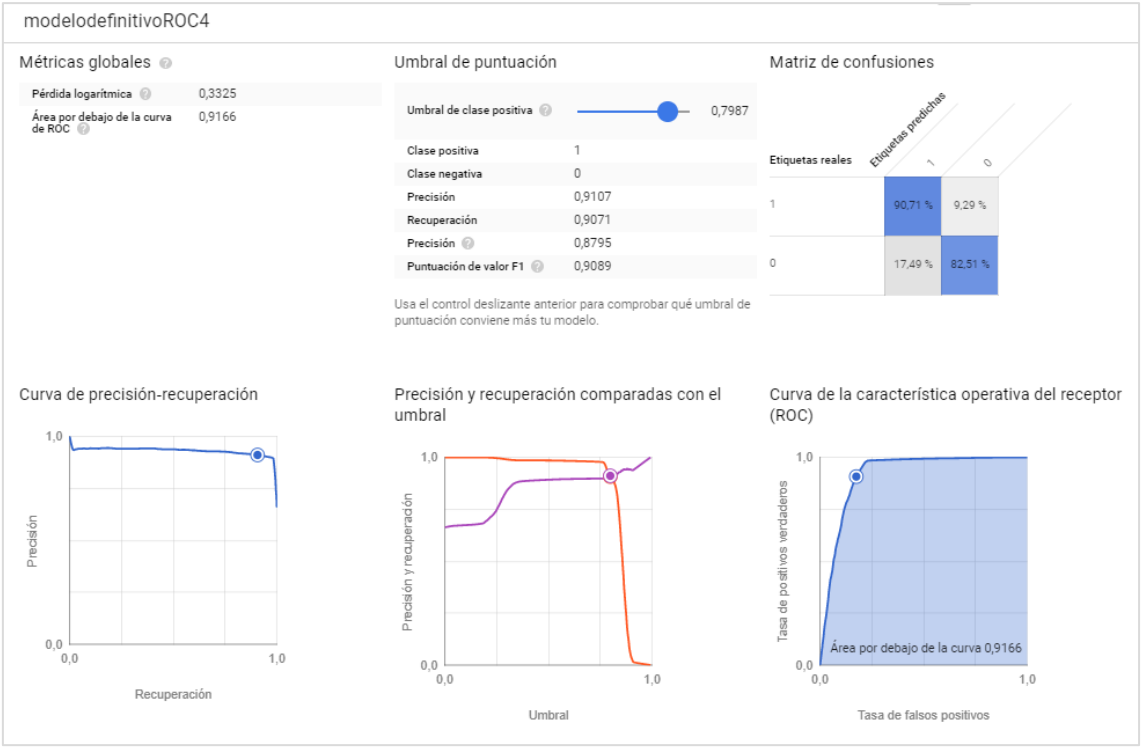
Modelo 3

Con estos resultados, decidimos parametrizar la consulta para seleccionar manualmente los conjuntos de train y test. Se una el parámetro data_split_method para seleccionar el modo ‘seq’, esto va a hacer que los conjuntos de datos se elijan según una columna ordenada, un index o una fecha, en mi caso escogí ‘date’. De esta forma, las observaciones se ordenan de menor a mayor y se selecciona una parte que corresponderá al test, el resto quedará para el train. En mi caso usé un común 70/30 para separar los conjuntos. Además de este avance, estudié las dimensiones que conformaban mi modelo2 y las pulí más:

Eliminé Campaña, fuente, navegador y añadí unas dimensiones personalizadas que usamos en DVS para medir todo tipo de aspectos de nuestro negocio. Estas son conexión, inapp e inventory. Conexión hace referencia a la conexión 3G o WIFI que usan los usuarios que se suscriben, de esta forma, un usuario que navega por 3G garantiza un plan de datos y por tanto saldo en su cuenta como para poder cobrar la suscripción. Inapp es una dimension que nos marca la aplicación en la cual se ha producido una transacción y por último el inventory. Esta dimension hace referencia al inventario que se compra en Google Ads. En este caso puede ser todo el tráfico 3G, todo el tráfico WIFI, todo el tráfico Orange, etc. La query quedó así:

```
CREATE OR REPLACE MODEL `wide-oasis-135923.iamarketingdvs.modelodefinitivoROC4`
OPTIONS(model_type='logistic_reg', data_split_method='SEQ', data_split_eval_fraction =
0.3, DATA_SPLIT_COL='date') AS
SELECT
IFNULL(SUM(( SELECT value FROM UNNEST(hits.customMetrics) WHERE index = 2)),0) AS label,
date, case when hits.hour > 7 and hits.hour < 13 then 'manana' when hits.hour > 13 and
hits.hour < 20 then 'tarde' else 'noche' end as momentodia, clientId, geoNetwork.city as
ciudad, (SELECT value FROM UNNEST(session.customDimensions) WHERE index = 14 GROUP BY 1)
AS producto, device.operatingSystem as OS, sum(totals.visits) as visitas, (SELECT value
FROM UNNEST(session.customDimensions) WHERE index = 1 GROUP BY 1) AS conexion, (SELECT
value FROM UNNEST(session.customDimensions) WHERE index = 4 GROUP BY 1) AS inapp,
(SELECT value FROM UNNEST(session.customDimensions) WHERE index = 19 GROUP BY 1) AS
inventory
FROM `wide-oasis-135923.140393857.ga_sessions_*` as session,
UNNEST(hits) AS hits
where _table_suffix BETWEEN '20190101' AND '20191001'
and trafficSource.campaign like '%range%'
and trafficSource.source like '%google%'
and totals.transactions > 0
GROUP BY date, clientId, ciudad, producto, OS, momentodia, conexion, inapp, inventory
```

Con esta evaluación:



Como se puede comprobar, el AUC es ahora > 0.9, un valor francamente bueno, especialmente viniendo de un 0.6 en el anterior modelo. Un AUC tan alto nos permite seleccionar un umbral con enormes posibilidades de crear verdaderos positivos sacrificando muy pocos falsos

positivos. De esta forma, elijo 0.7987 para conseguir un 90% de true positive y sacrificando sólo un 17% de false positive.

Vamos ahora a predecir resultados por OS sobre un día posterior al train/test y comparar los resultados con lo que se obtuvieron realmente. Este día será el 2 de octubre, el inmediatamente posterior al último de train/test.

La query queda de la siguiente manera:

```
SELECT
  OS, SUM(predicted_label) as totalBilledPredicted
FROM ML.PREDICT(MODEL `wide-oasis-135923.iamarketingdvs.modelodefinitivoROC4`, (
SELECT
IFNULL(SUM(( SELECT value FROM UNNEST(hits.customMetrics) WHERE index = 2)),0) AS label,
date, case when hits.hour > 7 and hits.hour < 13 then 'manana' when hits.hour > 13 and
hits.hour < 20 then 'tarde' else 'noche' end as momentodia, hits.hour as hora, clientId,
geoNetwork.city as ciudad, device.deviceCategory as categoriaDispositivo,
device.mobileDeviceBranding as marca, device.browser as navegador, trafficSource.source
as fuente, trafficSource.campaign as campana, (SELECT value FROM
UNNEST(session.customDimensions) WHERE index = 14 GROUP BY 1) AS producto,
device.operatingSystem as OS, sum(totals.visits) as visitas, (SELECT value FROM
UNNEST(session.customDimensions) WHERE index = 1 GROUP BY 1) AS conexion, (SELECT value
FROM UNNEST(session.customDimensions) WHERE index = 4 GROUP BY 1) AS inapp, (SELECT
value FROM UNNEST(session.customDimensions) WHERE index = 19 GROUP BY 1) AS inventory
FROM `wide-oasis-135923.140393857.ga_sessions_*` as session,
UNNEST(hits) AS hits
where _table_suffix BETWEEN '20191002' AND '20191002'
and trafficSource.campaign like '%range%'
and trafficSource.source like '%google%'
and totals.transactions > 0
GROUP BY date, clientId, ciudad, marca, categoriaDispositivo, navegador, campana,
fuente, producto, OS, hora, conexion, inapp, inventory),
    STRUCT(0.7978 AS threshold)) group by OS order by totalBilledPredicted desc
```

*Nótese como se aplica al final de la query el threshold que elegimos en la fase de evaluación, usando el control deslizante de la herramienta web de BQ ML: 0.7978.

Los resultados de la predicción son estos:

Fila	OS	totalBilledPredicted
1	Android	204
2	iOS	20

Los datos reales para este día son estos:

Fila	OS	realBilled
1	Android	191
2	iOS	22
3	Windows Phone	0
4	Linux	0
5	Macintosh	0
6	(not set)	0
7	Windows	0

Como podíamos esperar, una predicción bastante ajustada a lo que realmente se produjo realmente ese día.

Vamos a predecir resultados por marca del dispositivo:

La query es esta:

```
SELECT
  marca, SUM(predicted_label) as totalBilledPredicted
FROM ML.PREDICT(MODEL `wide-oasis-135923.iamarketingdvs.modelodefinitivoROC4`, (
```

```
SELECT
IFNULL(SUM(( SELECT value FROM UNNEST(hits.customMetrics) WHERE index = 2)),0) AS label,
date, case when hits.hour > 7 and hits.hour < 13 then 'manana' when hits.hour > 13 and
hits.hour < 20 then 'tarde' else 'noche' end as momentodia, hits.hour as hora, clientId,
geoNetwork.city as ciudad, device.deviceCategory as categoriaDispositivo,
device.mobileDeviceBranding as marca, device.browser as navegador, trafficSource.source
as fuente, trafficSource.campaign as campana, (SELECT value FROM
UNNEST(session.customDimensions) WHERE index = 14 GROUP BY 1) AS producto,
device.operatingSystem as OS, sum(totals.visits) as visitas, (SELECT value FROM
UNNEST(session.customDimensions) WHERE index = 1 GROUP BY 1) AS conexion, (SELECT value
FROM UNNEST(session.customDimensions) WHERE index = 4 GROUP BY 1) AS inapp, (SELECT
value FROM UNNEST(session.customDimensions) WHERE index = 19 GROUP BY 1) AS inventory
FROM `wide-oasis-135923.140393857.ga_sessions_*` as session,
UNNEST(hits) AS hits
where _table_suffix BETWEEN '20191002' AND '20191002'
and trafficSource.campaign like '%range%'
and trafficSource.source like '%google%'
and totals.transactions > 0
GROUP BY date, clientId, ciudad, marca, categoriaDispositivo, navegador, campana,
fuente, producto, OS, hora, conexion, inapp, inventory),
    STRUCT(0.7978 AS threshold)) group by marca order by totalBilledPredicted desc
```

El resultado de la predicción es este:

Fila	marca	totalBilledPredicted
1	Samsung	87
2	Huawei	52
3	Apple	20
4	LG	18
5	(not set)	12
6	Xiaomi	11
7	BQ	7
8	Motorola	5
9	Sony	2
10	Wiko	2
11	ZTE	2
12	LeEco	1

Los resultados reales fueron estos:

Fila	marca	realBilled
1	Samsung	84
2	Huawei	45
3	Apple	22
4	LG	16
5	(not set)	12
6	Xiaomi	10
7	BQ	8
8	Motorola	4
9	Wiko	3
10	Sony	2
11	OPPO	2
12	LeEco	1

Podemos confirmar que tenemos un modelo bastante bueno.

Lo que vamos a hacer ahora es sacar los billed predichos para el segundo modelo, para el tercer modelo y los reales para algunas features (coincidentes entre modelo 2 y 3), y vamos a hacer un estudio en Tableau comparándolo todo. Se puede ver en el archivo

modelos2y3_vs_reales_ROC. Esto generará 3 datasets que guardaré en 3 tablas de BQ a las cuales haré un join para analizar las tres métricas por cada una de las dimensiones elegidas, en este caso serán marca de dispositivo, sistema operativo, clientId y campaña de adwords.

Como comentarios, en la gráfica de firstBilled por clientId, se ve claramente como el modelo 3 acierta muchos más usuarios que el modelo 2. Lo he ordenado por billing reales para comprobar en que columna se ven mayor densidad de predictedBilled, la del modelo 2 o la del modelo 3, y gana como es normal el modelo 3.

3.- CONCLUSIONES

Se puedo concluir pues que en todas las comparaciones: por OS, marca de dispositivo, campaña y clientId, el modelo 3 se ajusta mucho más a la realidad que el modelo 2, esto sin embargo, no hacer más que confirmar que un modelo con un AUC tan superior a otro (0.9 vs 0.6), combinado con una buena elección de umbral de clase positiva, será siempre más eficiente a la hora de predecir resultados futuros.

El modelo finalmente obtenido sirve para predecir con una exactitud bastante aceptable los usuarios que serán cobrados y los que no de entre todos los que cursan una suscripción. Gracias a esto, vamos a poder trabajar en un futuro en crear audiencias desde GA que se usarán en AW para aplicarles estrategias de puja diferentes entre sí.

4.- DOCUMENTACION:

Sintaxis de en BQ ML:

https://cloud.google.com/bigquery-ml/docs/reference/standard-sql/bigqueryml-syntax-create

How to query and calculate Google Analytics Data in BigQuery

https://towardsdatascience.com/how-to-query-and-calculate-google-analytics-data-in-bigquery-cab8fc4f396#f2c8

Esquema raw de campos de BigQuery Export

https://support.google.com/analytics/answer/3437719?hl=es

Primeros pasos con BigQuery ML en la IU web

https://cloud.google.com/bigquery-ml/docs/bigqueryml-web-ui-start

Machine learning video tutorials

https://developers.google.com/machine-learning/crash-course/classification/video-lecture?hl=es-419

W3schools SQL Tutorial

https://www.w3schools.com/sql/default.asp

Lucas Díaz García - Diciembre 2019

Modelo de regresión logística en dataset de Google Analytics

TFM Data Science (Kschool)