FIAP Challenge Sofisa - Demonstração do Teradata Vantage

Cesar Augusto da Silva - educador.cesar@gmail.com

Lucas Alexandre Alves da Silva - lucas.alexandre1@hotmail.com

Marivaldo dos Santos Barbosa - marivaldo.barbosa@icloud.com

Thiago Alberto de Souza Colen - thiago.souzacolen@gmail.com

Como protótipo da utilização da arquitetura escolhida pela equipe, vamos demonstrar neste Jupyter Notebook um exemplo de utilização do Teradata Vantage facilitando o dia-a-dia do Cientista de Dados por meio de "levar o analítico ao dado" ao invés de "levar o dado ao analitico". Utilizaremos o Teradata SQL Kernel, disponível gratuitamente através do downloads.teradata.com. Com esse Kernel instalado, é possível realizar consultas no Teradata utilizando o mesmo JupyterHub que os usuários (Cientistas de Dados) já costumam a utilizar.

Neste exemplo, utilizaremos dados estruturados dentro do banco de dados Teradata Vantage, onde a partir do mesmo, habilitamos o Cientista de Dados à executar modelos de predição simples e que o auxiliará na criação de variáveis, de maneira eficaz e rápida, por não precisar levar o dado à outra plataforma.

Apresentamos então dois modelos de regressão, muito utilizados no setor financeiro, Regressão Linear e Regressão Logística.

Regressão Linear

A Regressão Linear é um dos tipos fundamentais de algoritmos de modelagem preditiva. Na regressão linear, uma variável numérica dependente é expressa em termos da soma de uma ou mais variáveis numéricas independentes, cada uma multiplicada por um coeficiente numérico, geralmente com um termo constante adicionado à soma das variáveis independentes. A regressão linear consiste nos coeficientes das variáveis independentes juntamente com um termo constante que compõe um modelo de regressão linear. A aplicação desses coeficientes às variáveis (colunas) de cada observação (linha) em um conjunto de dados (tabela) é conhecida como "Scoring".

Regressão Logística

A Regressão Logística é um dos tipos de análise estatística mais amplamente utilizados. Na Regressão Logística, um conjunto de variáveis independentes (neste caso colunas) é processado para prever o valor de uma variável dependente (coluna) que assume dois valores referidos como resposta (1) e não-resposta (0). Na verdade, o usuário especifica qual valor da variável dependente ele irá tratar como a resposta, e todos os outros valores assumidos pela variável dependente são tratados como não-resposta. O resultado não é, entretanto, uma variável numérica contínua como visto na Regressão Linear, mas sim uma probabilidade entre 0 e 1 de que o valor de resposta é assumido pela variável dependente.

1) Primeramente, vamos realizar o setup da variavel SystemName indicando a qual banco de dados Teradata iremos nos conectar.

In [1]:

```
%var SystemName=TRDT01
```

2) Em seguida, através da variável QLID, vamos identificar em qual database/usuário iremos armazenar os resultados de nossas consultas e predições.

In [2]:

```
%var QLID=NOS_USR
```

3) A variável a seguir indica onde estão localizadas as bibliotecas utilizadas para nossa modelagem. Nesse caso, para facilitar a demonstração, utilizamos o nosso mesmo usuário acima.

In [3]:

```
%var XSPDB=NOS_USR
```

Através do comando %connect abaixo, iremos nos conectar ao banco de dados definido anteriormente.

In [4]:

```
%connect ${SystemName}
```

```
Success: 'TRDT01' connection established and activated for user 'nos_usr'
```

Vamos definir o database default, de utilização, para onde estão localizadas as bibliotecas para modelagem.

In [5]:

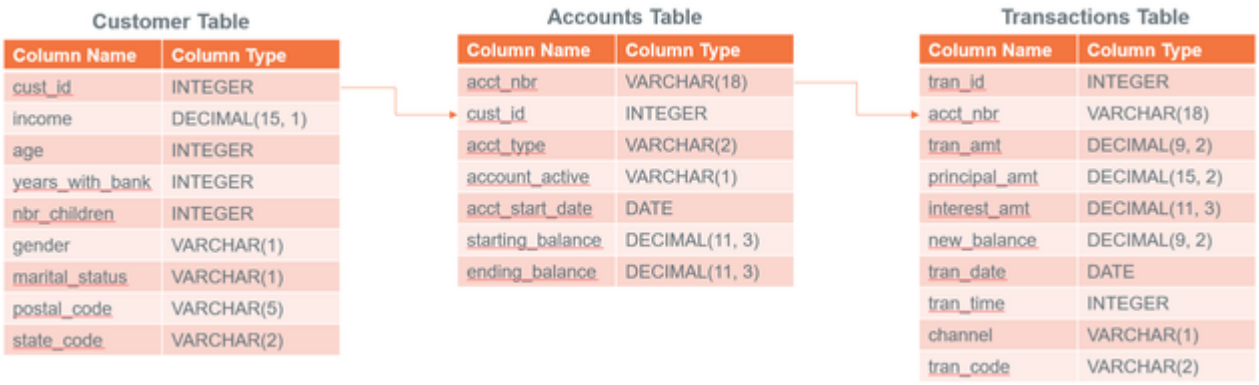
```
DATABASE ${XSPDB};
```

Out[5]:

```
Success: 1 rows affected
```

DataSet de Demonstração - Financeiro - Customers/Accounts/Transactions

Os dataset utilizado foi armazenado no nosso database de trabalho. É um dataset bancário fictício com a seguinte volumetria: tabela de clientes (customer) com ~10 mil linhas, tabela de contas (accounts) com ~100 mil linhas e tabela de transações (transactions) com ~1 milhão de linhas. Tais tabelas estão relacionadas seguindo o diagrama abaixo:



Customer Table

Column Name	Column Type
cust_id	INTEGER
income	DECIMAL(15, 1)
age	INTEGER
years_with_bank	INTEGER
nbr_children	INTEGER
gender	VARCHAR(1)
marital_status	VARCHAR(1)
postal_code	VARCHAR(5)
state_code	VARCHAR(2)

Accounts Table

Column Name	Column Type
acct_nbr	VARCHAR(18)
cust_id	INTEGER
acct_type	VARCHAR(2)
account_active	VARCHAR(1)
acct_start_date	DATE
starting_balance	DECIMAL(11, 3)
ending_balance	DECIMAL(11, 3)

Transactions Table

Column Name	Column Type
tran_id	INTEGER
acct_nbr	VARCHAR(18)
tran_amt	DECIMAL(9, 2)
principal_amt	DECIMAL(15, 2)
interest_amt	DECIMAL(11, 3)
new_balance	DECIMAL(9, 2)
tran_date	DATE
tran_time	INTEGER
channel	VARCHAR(1)
tran_code	VARCHAR(2)

(https://ibb.co/zQR4dTt)

In [6]:

```
SELECT * FROM ${XSPDB}.Customer SAMPLE 10;
```

Out[6]:

	cust_id	income	age	years_with_bank	nbr_children	gender	marital_status	name_pr
1	1363068	57612	58	8	2	M	2	Mr.
2	1362500	56708	40	0	0	M	1	
3	1363199	0	16	1	0	M	1	
4	1363088	28975	54	4	2	M	2	
5	1363213	38010	60	0	0	M	2	Mr.
6	1362720	18806	19	4	0	M	1	
7	1362805	4626	78	8	0	F	3	Ms.
8	1362941	18377	36	6	2	F	2	
9	1362920	5781	80	0	0	F	1	
10	1363247	19686	64	4	0	F	2	Ms.

In [7]:

```
%meta
```

```
Result Set ID: /home/jovyan/JupyterLabRoot/Teradata/Resultsets/2021.04.02_21.
36.10.568_UTC
History ID:    936
Rows:          10 of 10
Parts:         2
Column Definitions:
    cust_id: INTEGER
    income: INTEGER
    age: SMALLINT
    years_with_bank: SMALLINT
    nbr_children: SMALLINT
    gender: CHAR(1)
    marital_status: CHAR(1)
    name_prefix: CHAR(4)
    first_name: CHAR(30)
    last_name: CHAR(30)
    street_nbr: SMALLINT
    street_name: CHAR(30)
    postal_code: CHAR(5)
    city_name: CHAR(20)
    state_code: CHAR(2)
```

In [8]:

```
SELECT * FROM ${XSPDB}.Accounts SAMPLE 10;
```

Out[8]:

	acct_nbr	cust_id	acct_type	account_active	acct_start_date	acct_end_date
1	0000000013631122	1363112	CK	Y	1994-07-08	
2	0000000013632963	1363296	SV	Y	1991-06-28	
3	0000000013625723	1362572	SV	Y	1993-11-17	
4	4561143213633084	1363308	CC	N	1994-08-10	1995-04-15
5	4561143213626054	1362605	CC	Y	1991-01-11	
6	0000000013634652	1363465	CK	Y	1995-12-29	
7	0000000013633213	1363321	SV	Y	1994-07-23	
8	0000000013631943	1363194	SV	N	1990-09-25	1995-04-21
9	0000000013628483	1362848	SV	Y	1990-08-07	
10	0000000013625622	1362562	CK	Y	1995-11-17	

In [9]:

```
%meta
```

```
Result Set ID: /home/jovyan/JupyterLabRoot/Teradata/Resultsets/2021.04.02_21.
36.10.752_UTC
History ID:    937
Rows:          10 of 10
Parts:         2
Column Definitions:
    acct_nbr: CHAR(16)
    cust_id: INTEGER
    acct_type: CHAR(2)
    account_active: CHAR(1)
    acct_start_date: DATE
    acct_end_date: DATE
    starting_balance: DECIMAL(9, 2)
    ending_balance: DECIMAL(9, 2)
```

In [10]:

```
SELECT * FROM ${XSPDB}.Transactions SAMPLE 10;
```

Out[10]:

	tran_id	acct_nbr	tran_amt	principal_amt	interest_amt	new_balance	tran_d
1	1	4561143213634554	-157.23	-157.23	.00	-157.23	1995-1
2	80	0000000013633352	-.15	-.15	.00	696.63	1995-0
3	27	4561143213629454	288.66	286.91	1.75	-432.77	1995-0
4	50	0000000013625732	593.22	593.22	.00	6891.60	1995-0
5	31	4561143213632004	78.07	74.33	3.74	-191.39	1995-0
6	81	0000000013628772	.00	.00	.00	18.34	1995-0
7	6	0000000013627362	3.46	.00	3.46	2770.29	1995-0
8	10	0000000013634052	-171.19	-171.19	.00	143.33	1995-0
9	35	4561143213626634	333.41	287.99	45.42	-4112.01	1995-0
10	14	0000000013631572	-51.01	-51.01	.00	9480.22	1995-1

In [11]:

```
%meta
```

```
Result Set ID: /home/jovyan/JupyterLabRoot/Teradata/Resultsets/2021.04.02_21.
36.10.878_UTC
History ID:    938
Rows:          10 of 10
Parts:         2
Column Definitions:
    tran_id: INTEGER
    acct_nbr: CHAR(16)
    tran_amt: DECIMAL(9, 2)
    principal_amt: DECIMAL(9, 2)
    interest_amt: DECIMAL(9, 2)
    new_balance: DECIMAL(9, 2)
    tran_date: DATE
    tran_time: INTEGER
    channel: CHAR(1)
    tran_code: CHAR(2)
```

Criamos o seguinte Data Set Analítico (ADS), por meio do Join dessas 3 tabelas:

In [12]:

```
CREATE TABLE ${XSPDB}.VAL_ADS AS (
    SELECT
        T1.cust_id  AS cust_id
       ,MIN(T1.income) AS tot_income
       ,MIN(T1.age) AS tot_age
       ,MIN(T1.years_with_bank) AS tot_cust_years
       ,MIN(T1.nbr_children) AS tot_children
       ,CASE WHEN MIN(T1.marital_status) = 1 THEN 1 ELSE 0 END AS single_ind
       ,CASE WHEN MIN(T1.gender) = 'F' THEN 1 ELSE 0 END AS female_ind
       ,CASE WHEN MIN(T1.marital_status) = 2 THEN 1 ELSE 0 END AS married_ind
       ,CASE WHEN MIN(T1.marital_status) = 3 THEN 1 ELSE 0 END AS separated_ind
       ,MAX(CASE WHEN T1.state_code = 'CA' THEN 1 ELSE 0 END) AS ca_resident_ind
       ,MAX(CASE WHEN T1.state_code = 'NY' THEN 1 ELSE 0 END) AS ny_resident_ind
       ,MAX(CASE WHEN T1.state_code = 'TX' THEN 1 ELSE 0 END) AS tx_resident_ind
       ,MAX(CASE WHEN T1.state_code = 'IL' THEN 1 ELSE 0 END) AS il_resident_ind
       ,MAX(CASE WHEN T1.state_code = 'AZ' THEN 1 ELSE 0 END) AS az_resident_ind
       ,MAX(CASE WHEN T1.state_code = 'OH' THEN 1 ELSE 0 END) AS oh_resident_ind
       ,MAX(CASE WHEN T2.acct_type = 'CK' THEN 1 ELSE 0 END) AS ck_acct_ind
       ,MAX(CASE WHEN T2.acct_type = 'SV' THEN 1 ELSE 0 END) AS sv_acct_ind
       ,MAX(CASE WHEN T2.acct_type = 'CC' THEN 1 ELSE 0 END) AS cc_acct_ind
       ,AVG(CASE WHEN T2.acct_type = 'CK' THEN T2.starting_balance+T2.ending_balance ELSE
0 END) AS ck_avg_bal
       ,AVG(CASE WHEN T2.acct_type = 'SV' THEN T2.starting_balance+T2.ending_balance ELSE
0 END) AS sv_avg_bal
       ,AVG(CASE WHEN T2.acct_type = 'CC' THEN T2.starting_balance+T2.ending_balance ELSE
0 END) AS cc_avg_bal
       ,AVG(CASE WHEN T2.acct_type = 'CK' THEN T3.principal_amt+T3.interest_amt ELSE 0 END
) AS ck_avg_tran_amt
       ,AVG(CASE WHEN T2.acct_type = 'SV' THEN T3.principal_amt+T3.interest_amt ELSE 0 END
) AS sv_avg_tran_amt
       ,AVG(CASE WHEN T2.acct_type = 'CC' THEN T3.principal_amt+T3.interest_amt ELSE 0 END
) AS cc_avg_tran_amt
       ,COUNT(CASE WHEN ((EXTRACT(MONTH FROM T3.tran_date) + 2) / 3) = 1 THEN T3.tran_id E
LSE NULL END) AS q1_trans_cnt
       ,COUNT(CASE WHEN ((EXTRACT(MONTH FROM T3.tran_date) + 2) / 3) = 2 THEN T3.tran_id E
LSE NULL END) AS q2_trans_cnt
       ,COUNT(CASE WHEN ((EXTRACT(MONTH FROM T3.tran_date) + 2) / 3) = 3 THEN T3.tran_id E
LSE NULL END) AS q3_trans_cnt
       ,COUNT(CASE WHEN ((EXTRACT(MONTH FROM T3.tran_date) + 2) / 3) = 4 THEN T3.tran_id E
LSE NULL END) AS q4_trans_cnt
    FROM ${XSPDB}.Customer AS T1
        LEFT OUTER JOIN ${XSPDB}.Accounts AS T2
            ON T1.cust_id = T2.cust_id
        LEFT OUTER JOIN ${XSPDB}.Transactions AS T3
            ON T2.acct_nbr = T3.acct_nbr
GROUP BY T1.cust_id) WITH DATA UNIQUE PRIMARY INDEX (cust_id);
```

```
Unable to run SQL: Unable to run SQL query: Database reported error:3803:Tabl
e 'VAL_ADS' already exists.
```

In [13]:

```
SELECT * FROM ${XSPDB}.VAL_ADS SAMPLE 10;
```

Out[13]:

	cust_id	tot_income	tot_age	tot_cust_years	tot_children	single_ind	female_ind	ma
1	1363035	8262	60	0	0	0	0	1
2	1362705	27635	40	0	2	0	1	0
3	1363174	21384	58	7	1	0	1	1
4	1362668	0	13	5	0	1	0	0
5	1363133	12823	66	6	0	1	1	0
6	1362594	69205	53	3	2	0	0	1
7	1363259	7335	28	7	1	0	0	1
8	1362935	0	15	0	0	1	0	0
9	1362962	2858	83	3	0	0	0	0
10	1362541	22362	63	3	0	0	0	1

In [14]:

```
%meta
```

```
Result Set ID: /home/jovyan/JupyterLabRoot/Teradata/Resultsets/2021.04.02_21.
36.11.242_UTC
History ID:    940
Rows:          10 of 10
Parts:         2
Column Definitions:
    cust_id: INTEGER
    tot_income: INTEGER
    tot_age: SMALLINT
    tot_cust_years: SMALLINT
    tot_children: SMALLINT
    single_ind: BYTEINT
    female_ind: BYTEINT
    married_ind: BYTEINT
    separated_ind: BYTEINT
    ca_resident_ind: BYTEINT
    ny_resident_ind: BYTEINT
    tx_resident_ind: BYTEINT
    il_resident_ind: BYTEINT
    az_resident_ind: BYTEINT
    oh_resident_ind: BYTEINT
    ck_acct_ind: BYTEINT
    sv_acct_ind: BYTEINT
    cc_acct_ind: BYTEINT
    ck_avg_bal: FLOAT(0, 0)
    sv_avg_bal: FLOAT(0, 0)
    cc_avg_bal: FLOAT(0, 0)
    ck_avg_tran_amt: FLOAT(0, 0)
    sv_avg_tran_amt: FLOAT(0, 0)
    cc_avg_tran_amt: FLOAT(0, 0)
    q1_trans_cnt: INTEGER
    q2_trans_cnt: INTEGER
    q3_trans_cnt: INTEGER
    q4_trans_cnt: INTEGER
```

Regressão Linear

1. Utilizando o dataset analítico criado (tabela VAL_ADS) vamos construir um modelo linear para estimar o saldo médio mensal (cc_avg_bal) que um cliente bancário tem em seu cartão de crédito com base em todas as variáveis não relacionadas a cartão de crédito na tabela. Os coeficientes e estatísticas de variáveis de modelo são criadas na tabela LinearRegressionDemo1 conforme especificado pelo argumento outputtablename. Observe que as estatísticas do modelo são criadas na tabela LinearRegressionDemo1_rpt.

In [15]:

```
call ${XSPDB}.td_analyze('linear',
                         'database=${XSPDB};
                          tablename=VAL_ADS;
                          columns=tot_age,tot_income,tot_cust_years,tot_children,single_in
d,female_ind,married_ind,separated_ind,ck_acct_ind,sv_acct_ind,sv_avg_bal,ck_avg_bal,ca_re
sident_ind,ny_resident_ind,tx_resident_ind,il_resident_ind,az_resident_ind,oh_resident_ind
;
                          dependent=cc_avg_bal;
                          neardependencyreport=true;
                          outputdatabase=${QLID};
                          outputtablename=LinearRegressionDemo1');
```

Out[15]:

```
Success: 0 rows affected
```

In [16]:

```
SELECT * FROM ${QLID}.LinearRegressionDemo1 order by 2 DESC;
```

Out[16]:

	Column Name	B Coefficient	Standard Error	T Statistic	
1	separated_ind	328.174825176008	169.342592085858	1.93793434441833	0.053
2	(Constant)	141.038765533841	197.290249312991	0.714879554488726	0.474
3	female_ind	97.6713334824149	66.9278782111552	1.4593520083554	0.144
4	oh_resident_ind	53.9832449662958	193.46160098053	0.279038551798859	0.780
5	single_ind	39.4868288639353	145.456747470657	0.271467838725743	0.786
6	tot_cust_years	19.447927470585	12.5205484386463	1.55328079803249	0.120
7	tot_age	2.20096107482116	2.28116113917766	0.964842437923779	0.334
8	tot_income	0.0161884534218535	0.00177005055701053	9.14575764954111	0
9	sv_avg_bal	-0.02764065651019	0.0361464293578714	-0.764685668853508	0.444
10	ck_avg_bal	-0.0369196825088095	0.0178219709569634	-2.07158246402507	0.038
11	tot_children	-0.0643644852171628	39.172089504306	-0.0016431210597047	0.998
12	sv_acct_ind	-13.4974414493949	73.5267929246847	-0.1835717418441	0.854
13	ny_resident_ind	-16.4549210612285	101.425633792693	-0.162236314883289	0.871
14	ca_resident_ind	-44.0257596511528	86.7611601465814	-0.507436271907523	0.612
15	il_resident_ind	-46.7034550355164	131.663139950911	-0.354719286300853	0.722
16	married_ind	-53.1178586570904	120.066160995695	-0.442404905900129	0.658
17	ck_acct_ind	-128.013266444814	83.1779243231198	-1.53902934566536	0.124
18	tx_resident_ind	-141.265909945983	111.864519358269	-1.26283034832117	0.207
19	az_resident_ind	-282.342277438546	189.893518881843	-1.48684525465152	0.137

In [17]:

```
SELECT * FROM ${QLID}.LinearRegressionDemo1_rpt order by 2 DESC;
```

Out[17]:

	rid	Total Observations	Total Sum of Squares	Multiple Correlation Coefficient (R):	Squared Multiple Correlation Coefficient (1-Tolerance)	Ad
1	1	747	6.72013534623019e+08	0.385750211423687	0.148803225613419	0.127

In [18]:

```
SELECT * FROM ${QLID}.LinearRegressionDemo1_txt order by 2 DESC;
```

Out[18]:

	partId	XmlModel
1	1	NOS_USRVAL_ADS18cc_avg_balIncludeConstantnone

In [19]:

```
SELECT XMLSERIALIZE(Content X.Dot) as XMLText
FROM (SELECT * FROM ${QLID}.LinearRegressionDemo1_txt) AS C,
XMLTable (
'//*'
PASSING CREATEXML(C.XmlModel)
) AS X ("Dot");
```

Out[19]:

	XMLText
1	NOS_USRVAL_ADS18cc_avg_balIncludeConstantnone
2	NOS_USRVAL_ADS18cc_avg_balIncludeConstantnone
3	NOS_USR
4	VAL_ADS
5	18
6	cc_avg_bal
7	IncludeConstant
8	none
9	

2) A fim de mostrar o recurso "GROUP BY" da Regressão Linear, vamos construir um modelo para cada state_code; para fazer isso, o dataset analítico original foi modificado para incluir state_code em vez das variáveis indicadoras dos estados:

In [20]:

```
CREATE TABLE ${XSPDB}.VAL_ADS2 AS (
    SELECT
        T1.cust_id  AS cust_id
       ,MIN(T1.income) AS tot_income
       ,MIN(T1.age) AS tot_age
       ,MIN(T1.years_with_bank) AS tot_cust_years
       ,MIN(T1.nbr_children) AS tot_children
       ,CASE WHEN MIN(T1.marital_status) = 1 THEN 1 ELSE 0 END AS single_ind
       ,CASE WHEN MIN(T1.gender) = 'F' THEN 1 ELSE 0 END AS female_ind
       ,CASE WHEN MIN(T1.marital_status) = 2 THEN 1 ELSE 0 END AS married_ind
       ,CASE WHEN MIN(T1.marital_status) = 3 THEN 1 ELSE 0 END AS separated_ind
       ,MAX(CASE WHEN T1.state_code = 'CA' THEN 'CA'
                 WHEN T1.state_code = 'NY' THEN 'NY'
                 WHEN T1.state_code = 'TX' THEN 'TX'
                 WHEN T1.state_code = 'IL' THEN 'IL'
                 WHEN T1.state_code = 'AZ' THEN 'AZ'
                 WHEN T1.state_code = 'OH' THEN 'OH' ELSE 'OTHER' END) AS state_code
       ,MAX(CASE WHEN T2.acct_type = 'CK' THEN 1 ELSE 0 END) AS ck_acct_ind
       ,MAX(CASE WHEN T2.acct_type = 'SV' THEN 1 ELSE 0 END) AS sv_acct_ind
       ,MAX(CASE WHEN T2.acct_type = 'CC' THEN 1 ELSE 0 END) AS cc_acct_ind
       ,AVG(CASE WHEN T2.acct_type = 'CK' THEN T2.starting_balance+T2.ending_balance ELSE
0 END) AS ck_avg_bal
       ,AVG(CASE WHEN T2.acct_type = 'SV' THEN T2.starting_balance+T2.ending_balance ELSE
0 END) AS sv_avg_bal
       ,AVG(CASE WHEN T2.acct_type = 'CC' THEN T2.starting_balance+T2.ending_balance ELSE
0 END) AS cc_avg_bal
       ,AVG(CASE WHEN T2.acct_type = 'CK' THEN T3.principal_amt+T3.interest_amt ELSE 0 END
) AS ck_avg_tran_amt
       ,AVG(CASE WHEN T2.acct_type = 'SV' THEN T3.principal_amt+T3.interest_amt ELSE 0 END
) AS sv_avg_tran_amt
       ,AVG(CASE WHEN T2.acct_type = 'CC' THEN T3.principal_amt+T3.interest_amt ELSE 0 END
) AS cc_avg_tran_amt
       ,COUNT(CASE WHEN ((EXTRACT(MONTH FROM T3.tran_date) + 2) / 3) = 1 THEN T3.tran_id E
LSE NULL END) AS q1_trans_cnt
       ,COUNT(CASE WHEN ((EXTRACT(MONTH FROM T3.tran_date) + 2) / 3) = 2 THEN T3.tran_id E
LSE NULL END) AS q2_trans_cnt
       ,COUNT(CASE WHEN ((EXTRACT(MONTH FROM T3.tran_date) + 2) / 3) = 3 THEN T3.tran_id E
LSE NULL END) AS q3_trans_cnt
       ,COUNT(CASE WHEN ((EXTRACT(MONTH FROM T3.tran_date) + 2) / 3) = 4 THEN T3.tran_id E
LSE NULL END) AS q4_trans_cnt
    FROM ${XSPDB}.Customer AS T1
        LEFT OUTER JOIN ${XSPDB}.Accounts AS T2
            ON T1.cust_id = T2.cust_id
        LEFT OUTER JOIN ${XSPDB}.Transactions AS T3
            ON T2.acct_nbr = T3.acct_nbr
GROUP BY T1.cust_id) WITH DATA UNIQUE PRIMARY INDEX (cust_id);
```

```
Unable to run SQL: Unable to run SQL query: Database reported error:3803:Tabl
e 'VAL_ADS2' already exists.
```

In [21]:

```
SELECT * FROM ${XSPDB}.VAL_ADS2 SAMPLE 10;
```

Out[21]:

	cust_id	tot_income	tot_age	tot_cust_years	tot_children	single_ind	female_ind	ma
1	1362837	22322	50	0	0	0	0	1
2	1362811	8011	82	2	0	0	1	1
3	1362969	75121	58	8	0	0	1	0
4	1362745	11410	82	2	0	0	0	1
5	1362643	6356	18	3	0	1	1	0
6	1362980	8201	18	3	0	1	0	0
7	1362939	47267	54	4	1	0	0	0
8	1363270	0	16	1	0	1	0	0
9	1362650	17804	21	0	1	0	0	1
10	1363126	58749	50	0	2	0	1	1

In [22]:

```
%meta
```

```
Result Set ID: /home/jovyan/JupyterLabRoot/Teradata/Resultsets/2021.04.02_21.
36.19.051_UTC
History ID:    947
Rows:          10 of 10
Parts:         2
Column Definitions:
    cust_id: INTEGER
    tot_income: INTEGER
    tot_age: SMALLINT
    tot_cust_years: SMALLINT
    tot_children: SMALLINT
    single_ind: BYTEINT
    female_ind: BYTEINT
    married_ind: BYTEINT
    separated_ind: BYTEINT
    state_code: VARCHAR(5)
    ck_acct_ind: BYTEINT
    sv_acct_ind: BYTEINT
    cc_acct_ind: BYTEINT
    ck_avg_bal: FLOAT(0, 0)
    sv_avg_bal: FLOAT(0, 0)
    cc_avg_bal: FLOAT(0, 0)
    ck_avg_tran_amt: FLOAT(0, 0)
    sv_avg_tran_amt: FLOAT(0, 0)
    cc_avg_tran_amt: FLOAT(0, 0)
    q1_trans_cnt: INTEGER
    q2_trans_cnt: INTEGER
    q3_trans_cnt: INTEGER
    q4_trans_cnt: INTEGER
```

Agora, vamos construir um modelo de regressão linear para cada state_code. Os coeficientes e estatísticas de variáveis são criadas na tabela LinearRegressionDemo2 conforme especificado pelo argumento outputtablename. Observe que as estatísticas do modelo são criadas com a tabela LinearRegressionDemo2_rpt, uma para cada state_code.

In [23]:

```
call ${XSPDB}.td_analyze('linear',
                         'database=${XSPDB};
                          tablename=VAL_ADS2;
                          columns=tot_age,tot_income,tot_cust_years,tot_children,single_in
d,married_ind,separated_ind,female_ind,ck_acct_ind,sv_acct_ind,sv_avg_bal,ck_avg_bal;
                          dependent=cc_avg_bal;
                          outputdatabase=${QLID};
                          outputtablename=LinearRegressionDemo2;
                          groupby=state_code');
```

Out[23]:

```
Success: 0 rows affected
```

In [24]:

```
SELECT * FROM ${QLID}.LinearRegressionDemo2 ORDER BY 1, 2;
```

Out[24]:

	state_code	Column Name	B Coefficient	
1	AZ	(Constant)	-399.09910392595	807
2	AZ	ck_acct_ind	19.6571842074301	299
3	AZ	ck_avg_bal	0.0637643947858046	0.10
4	AZ	female_ind	241.818516844219	303
5	AZ	married_ind	231.337515030165	443
6	AZ	separated_ind	49.2832789069808	560
7	AZ	single_ind	-39.2395923720176	514
8	AZ	sv_acct_ind	-353.534424863496	353
9	AZ	sv_avg_bal	0.21018575008192	0.13
10	AZ	tot_age	10.9680673369554	11.5
11	AZ	tot_children	108.139628141342	236
12	AZ	tot_cust_years	40.5773396565698	41.5
13	AZ	tot_income	-0.00769643511966645	0.00
14	CA	(Constant)	639.198507549869	402
15	CA	ck_acct_ind	-424.256483587759	200
16	CA	ck_avg_bal	-0.0884927796524282	0.04
17	CA	female_ind	292.803241746662	153
18	CA	married_ind	-131.855844121547	241
19	CA	separated_ind	219.860319801548	389
20	CA	single_ind	-161.491665519527	304
21	CA	sv_acct_ind	-79.3766776461317	162
22	CA	sv_avg_bal	-0.114710883661639	0.06
23	CA	tot_age	0.312540412412478	5.62
24	CA	tot_children	-149.641695081813	77.5
25	CA	tot_cust_years	-23.6339691606112	28.5

In [25]:

```
SELECT * FROM ${QLID}.LinearRegressionDemo2_rpt ORDER BY 1, 2;
```

Out[25]:

	rid	state_code	Total Observations	Total Sum of Squares	Multiple Correlation Coefficient (R):	Squared Mult Correlatio Coefficient Tolerance
1	1	AZ	24	4.89443127840392e+06	0.67857381597088	0.4604624237
2	2	CA	177	2.0861452150148e+08	0.517933772968521	0.2682553931
3	3	IL	56	3.37652557123531e+07	0.658352657409312	0.433428221
4	4	NY	107	6.40704308897399e+07	0.462089029255775	0.2135262709
5	5	OH	23	9.29443841076735e+06		
6	6	OTHER	277	3.10330884847684e+08	0.434794289217757	0.1890460739
7	7	TX	83	3.91038748185278e+07	0.485821176169109	0.236022215

In [26]:

```
SELECT * FROM ${QLID}.LinearRegressionDemo2_txt ORDER BY 1, 2;
```

Out[26]:

	state_code	partId	XmlModel
1	AZ	1	NOS_USRVAL_ADS212cc_avg_ballncludeConstantnone
2	CA	1	NOS_USRVAL_ADS212cc_avg_ballncludeConstantnone
3	IL	1	NOS_USRVAL_ADS212cc_avg_ballncludeConstantnone
4	NY	1	NOS_USRVAL_ADS212cc_avg_ballncludeConstantnone
5	OH	1	NOS_USRVAL_ADS212cc_avg_ballncludeConstantnoneConstant columns d
6	OTHER	1	NOS_USRVAL_ADS212cc_avg_ballncludeConstantnone
7	TX	1	NOS_USRVAL_ADS212cc_avg_ballncludeConstantnone

In [27]:

```
SELECT XMLSERIALIZE(Content X.Dot) as XMLText
FROM (SELECT * FROM ${QLID}.LinearRegressionDemo2_txt) AS C,
XMLTable (
'//*'
PASSING CREATEXML(C.XmlModel)
) AS X ("Dot");
```

Out[27]:

	XMLText
1	NOS_USRVAL_ADS212cc_avg_balIncludeConstantnone
2	NOS_USRVAL_ADS212cc_avg_balIncludeConstantnone
3	NOS_USRVAL_ADS212cc_avg_balIncludeConstantnoneConstant columns detected...run tern
4	NOS_USRVAL_ADS212cc_avg_balIncludeConstantnone
5	NOS_USRVAL_ADS212cc_avg_balIncludeConstantnone
6	NOS_USRVAL_ADS212cc_avg_balIncludeConstantnone
7	NOS_USRVAL_ADS212cc_avg_balIncludeConstantnone
8	NOS_USRVAL_ADS212cc_avg_balIncludeConstantnoneConstant columns detected...run tern
9	NOS_USRVAL_ADS212cc_avg_balIncludeConstantnone
10	NOS_USRVAL_ADS212cc_avg_balIncludeConstantnone
11	NOS_USRVAL_ADS212cc_avg_balIncludeConstantnone
12	NOS_USRVAL_ADS212cc_avg_balIncludeConstantnone
13	NOS_USRVAL_ADS212cc_avg_balIncludeConstantnoneConstant columns detected...run tern
14	NOS_USRVAL_ADS212cc_avg_balIncludeConstantnone
15	NOS_USRVAL_ADS212cc_avg_balIncludeConstantnone
16	NOS_USRVAL_ADS212cc_avg_balIncludeConstantnone
17	NOS_USRVAL_ADS212cc_avg_balIncludeConstantnone
18	NOS_USRVAL_ADS212cc_avg_balIncludeConstantnoneConstant columns detected...run tern
19	NOS_USRVAL_ADS212cc_avg_balIncludeConstantnone
20	NOS_USRVAL_ADS212cc_avg_balIncludeConstantnone
21	NOS_USRVAL_ADS212cc_avg_balIncludeConstantnone
22	NOS_USRVAL_ADS212cc_avg_balIncludeConstantnone
23	NOS_USRVAL_ADS212cc_avg_balIncludeConstantnoneConstant columns detected...run tern
24	NOS_USRVAL_ADS212cc_avg_balIncludeConstantnone
25	NOS_USRVAL_ADS212cc_avg_balIncludeConstantnone
26	NOS_USRVAL_ADS212cc_avg_balIncludeConstantnone
27	NOS_USRVAL_ADS212cc_avg_balIncludeConstantnone
28	NOS_USRVAL_ADS212cc_avg_balIncludeConstantnoneConstant columns detected...run tern

Scoring da Regressão Linear

Objetivo

O Scoring da Regressão Linear é a aplicação de um modelo de regressão linear a uma tabela de entrada que contém as mesmas colunas de variáveis independentes contidas no modelo. O resultado é uma tabela de Scoring de saída que contém, no mínimo, uma ou mais colunas-chave e uma estimativa da variável dependente no modelo. O usuário também pode optar por realizar a avaliação do modelo, separadamente ou em combinação com o Scoring. Quando solicitado, um relatório é produzido como um conjunto de dados de resultados contendo o erro padrão da estimativa, bem como o erro absoluto mínimo, máximo e médio. Quando a avaliação do modelo é solicitada, a tabela de entrada deve contém uma coluna que representa a variável dependente no modelo. Quando tanto o Scoring quanto a avaliação são solicitadas, a tabela de saída inclui automaticamente o valor residual, calculado como a diferença entre o valor original e o valor previsto da variável dependente. O valor residual também pode ser solicitado quando apenas o Scoring é realizado.

---

1) Primeiramente, vamos executar o Scoring do modelo de regressão linear criado acima - para fins de demonstração, usaremos a mesma tabela VAL_ADS para o Scoring. O erro absoluto mínimo, máximo e médio e o erro padrão da estimativa são retornados como um conjunto de resultados. O conjunto de dados pós-Scoring inclui o real junto com a previsão e o residual que estamos avaliando também.

In [28]:

```
call ${XSPDB}.td_analyze('linearscore',
                         'database=${XSPDB};
                          tablename=VAL_ADS;
                          modeldatabase=${QLID};
                          modeltablename=LinearRegressionDemo1;
                          outputdatabase=${QLID};
                          outputtablename=LinearRegressionScore1;
                          predicted=estimate;
                          retain=cc_avg_bal;
                          scoringmethod=scoreandevaluate;');
```

Out[28]:

```
Success: 0 rows affected
```

In [29]:

```
SELECT * FROM ${QLID}.LinearRegressionScore1;
```

Out[29]:

	cust_id	cc_avg_bal	estimate	Residual
1	1363160	553.4672413793104	757.4626521344095	-203.99541075509907
2	1362813	0	567.6544791988415	-567.6544791988415
3	1362691	0	709.6684461677008	-709.6684461677008
4	1362588	421.9693548387097	305.28982833796016	116.67952650074953
5	1362752	287.7675	1024.122214403084	-736.3547144030839
6	1363261	317.9313368983957	535.3765804780354	-217.4452435796397
7	1362487	481.00349514563106	103.68281838174167	377.3206767638894
8	1362548	1163.5072180451127	924.8430201784508	238.66419786666194
9	1362996	0	695.1770295288964	-695.1770295288964
10	1363078	1009.2228571428572	307.26944186890796	701.9534152739492
11	1363200	67.59502824858757	348.44419087120673	-280.8491626226192
12	1362853	185.16379310344828	436.0954680841117	-250.93167498066342
13	1363017	338.7096774193548	-23.71646363002752	362.42614104938235
14	1363179	0	410.51851462068515	-410.51851462068515
15	1363343	0	-13.406086131950072	13.406086131950072
16	1362609	1207.4889655172415	346.62321538304695	860.8657501341945
17	1363158	73.80911392405064	467.70206920778264	-393.892955283732
18	1362670	0	300.7887628901124	-300.7887628901124
19	1363465	0	98.7932145851481	-98.7932145851481
20	1362506	4216.53	1428.9074690808181	2787.622530919182
21	1363486	0	994.6748147040951	-994.6748147040951
22	1362975	268.2368	750.8822876615538	-482.6454876615538
23	1363097	626.4476470588236	585.5249972779718	40.922649780851735
24	1362935	0	213.5400105200937	-213.5400105200937
25	1363198	0	1256.698607735741	-1256.698607735741
26	1363404	0	134.7274690123484	-134.7274690123484
27	1363322	0	1552.3245899881274	-1552.3245899881274
28	1362893	0	301.15734956638306	-301.15734956638306

2) Em seguida, vamos executar o Scoring dos modelos de regressão linear dos diversos state_code novamente usando a tabela VAL_ADS2. Para cada modelo, o erro absoluto mínimo, máximo e médio e o erro padrão da estimativa são retornados como um conjunto de resultados. O conjunto de dados pós-Scoring inclui o real junto com o state_code, a predição e o residual que estamos avaliando também.

In [30]:

```
call ${XSPDB}.td_analyze('linearscore',
                         'database=${XSPDB};
                          tablename=VAL_ADS2;
                          modeldatabase=${QLID};
                          modeltablename=LinearRegressionDemo2;
                          outputdatabase=${QLID};
                          outputtablename=LinearRegressionScore2;
                          predicted=estimate;
                          retain=cc_avg_bal;
                          scoringmethod=scoreandevaluate;');
```

Out[30]:

```
Success: 0 rows affected
```

In [31]:

```
SELECT * FROM ${QLID}.LinearRegressionScore2;
```

Out[31]:

	cust_id	state_code	cc_avg_bal	estimate	Residual
1	1363160	CA	553.4672413793104	952.9294944568617	-399.4622530775513
2	1362813	IL	0	760.5849494006503	-760.5849494006503
3	1362691	OTHER	0	756.4062612226761	-756.4062612226761
4	1362588	OTHER	421.9693548387097	269.4058033604687	152.5635514782409
5	1362752	CA	287.7675	1322.2094156863052	-1034.441915686305
6	1363261	NY	317.9313368983957	521.9640115676227	-204.032674669227
7	1362487	CA	481.00349514563106	-38.336865708376294	519.3403608540074
8	1362548	NY	1163.5072180451127	1385.4601749530248	-221.9529569079121
9	1362996	OTHER	0	600.9025290538956	-600.9025290538956
10	1363078	OTHER	1009.2228571428572	431.9215287585205	577.3013283843367
11	1363200	CA	67.59502824858757	203.32430921724776	-135.7292809686602
12	1362853	OTHER	185.16379310344828	302.4328747687115	-117.2690816652632
13	1363017	CA	338.7096774193548	-251.1334720360401	589.843149455395
14	1363179	IL	0	633.6170957100551	-633.6170957100551
15	1363343	TX	0	435.5721657117934	-435.5721657117934
16	1362609	OTHER	1207.4889655172415	201.7264368495738	1005.762528667667
17	1363158	OH	73.80911392405064		
18	1362670	CA	0	203.48926021385984	-203.4892602138598
19	1363465	CA	0	-106.03747080828055	106.0374708082805
20	1362506	CA	4216.53	2173.3720872875315	2043.157912712468
21	1363486	OTHER	0	1025.8597742264596	-1025.859774226459
22	1362975	OTHER	268.2368	1000.0518601218674	-731.8150601218674
23	1363097	OTHER	626.4476470588236	479.97037558819784	146.4772714706257
24	1362935	OTHER	0	55.94797567235012	-55.94797567235012
25	1363198	CA	0	1542.525615943326	-1542.525615943326
26	1363404	OH	0		
27	1363322	OTHER	0	1404.0068721998064	-1404.006872199806

Regressão Logística

1) Mais uma vez usando a tabela VAL_ADS, vamos construir um modelo de regressão logística para prever a propensão da base de clientes para abrir uma conta de cartão de crédito (cc_acct_ind) com base em todas as variáveis que não são de cartão de crédito no conjunto de dados analíticos. Os coeficientes do modelo e as estatísticas de variáveis são criados no outputtablename especificado. Além disso, as estatísticas do modelo são criadas dentro de uma tabela com uma extensão "_rpt" no outputtablename. Os relatórios para successtabpe, thresholdtable e lifttable são retornados em uma string XML dentro de uma tabela com uma extensão "_txt" no outputtablename.

In [32]:

```
call ${XSPDB}.td_analyze('logistic',
                         'database=${XSPDB};
                          tablename=VAL_ADS;
                          columns=tot_age,tot_income,tot_cust_years,tot_children,single_in
d,married_ind,separated_ind,female_ind,ck_acct_ind,sv_acct_ind,sv_avg_bal,ck_avg_bal,ca_re
sident_ind,ny_resident_ind,tx_resident_ind,il_resident_ind,az_resident_ind,oh_resident_ind
;
                          dependent=cc_acct_ind;
                          outputdatabase=${QLID};
                          outputtablename=LogisticOut1;
                          statstable=true;
                          successtable=true;
                          thresholdtable=true;
                          lifttable=true;');
```

Out[32]:

```
Success: 0 rows affected
```

In [33]:

```
SELECT * FROM ${QLID}.LogisticOut1 ORDER BY 2 DESC;
```

Out[33]:

	Column Name	B Coefficient	Standard Error	Wald Statistic	
1	ck_acct_ind	1.52093642910134	0.21122470736138	51.8480813763378	7.20
2	sv_acct_ind	1.00749545826154	0.194358930436256	26.8705894294096	5.18
3	separated_ind	0.875270852091336	0.5348948201149	2.67761509418834	1.63
4	oh_resident_ind	0.474759765128598	0.531936018276908	0.796579299625616	0.89
5	ny_resident_ind	0.368567190559411	0.276918289745215	1.77145474884433	1.33
6	female_ind	0.344845850408636	0.175064391622608	3.88020232073237	1.96
7	single_ind	0.287476775640691	0.39671872583054	0.525097716185238	0.72
8	tot_children	0.116107049904922	0.105336704651888	1.21494809767887	1.10
9	il_resident_ind	0.0615990756599514	0.347565379755017	0.031410537563781	0.17
10	tot_cust_years	0.0321964569098754	0.0336141845271347	0.917425916218723	0.95
11	tot_age	0.0120081431977366	0.00607197841582009	3.91103118008513	1.97
12	tot_income	2.20581467083245e-05	5.3264111504386e-06	17.1501794179416	4.14
13	ck_avg_bal	-0.000154293882900153	4.72644979043383e-05	10.6568123731273	-3.2
14	sv_avg_bal	-0.000411459591405243	0.000112427809459119	13.3938947849454	-3.6
15	married_ind	-0.199036028918138	0.324701194628088	0.375747088004724	-0.6
16	ca_resident_ind	-0.203825906734878	0.225422627557536	0.817567747754025	-0.9
17	tx_resident_ind	-0.413944509514016	0.291274320540961	2.01966740912539	-1.4
18	az_resident_ind	-0.471059230176051	0.49268863640861	0.914125754202504	-0.9
19	(Constant)	-2.06933528045715	0.535393530392961	14.9387922083998	-3.8

In [34]:

```
SELECT * FROM ${QLID}.LogisticOut1_rpt ORDER BY 1;
```

Out[34]:

	rid	Total Observations	Total Iterations	Initial Log Likelihood	Final Log Likelihood	Likelihood Ratio Test G Statistic

In [35]:

```
SELECT * FROM ${QLID}.LogisticOut1_txt ORDER BY 1;
```

Out[35]:

	partId	XmlModel
1	1	NOS_USRVAL_ADS18cc_acct_ind1none0falseIncludeConstanttot_age42.4792503346720

In [36]:

```
SELECT XMLSERIALIZE(Content X.Dot) as XMLText
FROM (SELECT * FROM ${QLID}.LogisticOut1_txt) AS C,
XMLTable (
'//*'
PASSING CREATEXML(C.XmlModel)
) AS X ("Dot");
```

Out[36]:

	XMLText
1	NOS_USRVAL_ADS18cc_acct_ind1none0falseIncludeConstanttot_age42.479250334672024
2	NOS_USRVAL_ADS18cc_acct_ind1none0falseIncludeConstanttot_age42.479250334672024
3	NOS_USR
4	VAL_ADS
5	18
6	cc_acct_ind
7	1
8	none
9	0
10	false
11	IncludeConstant
12	tot_age42.47925033467202419.11487876817342tot_income22728.28112449799322207.2214
13	tot_age42.47925033467202419.11487876817342
14	tot_age
15	42.479250334672024
16	19.11487876817342
17	tot_income22728.28112449799322207.221405394437
18	tot_income
19	22728.281124497993
20	22207.221405394437
21	tot_cust_years3.90763052208835362.675633899491365
22	tot_cust_years
23	3.9076305220883536
24	2.675633899491365
25	tot_children0.7148594377510041.1034099354173406
26	tot_children
27	0.714859437751004
28	1.1034099354173406

2) Agora, usando a tabela VAL_ADS2, vamos construir um modelo de regressão logística para prever a propensão da base de clientes para abrir uma conta de cartão de crédito (cc_acct_ind) com base em todas as variáveis que não são de cartão de crédito no conjunto de dados analíticos, criando um modelo para cada state_code usando a opção GROUPBY = state_code. Mais uma vez, os coeficientes do modelo e as estatísticas da variável são criados no outputtablename especificado para cada modelo state_code. Além disso, as estatísticas do modelo são criadas dentro de uma tabela com uma extensão "_rpt" no outputtablename. Os relatórios para successtabpe, thresholdtable e lifttable são retornados em uma string XML dentro de uma tabela com uma extensão "_txt" no outputtablename.

In [37]:

```
call ${XSPDB}.td_analyze('logistic',
                         'database=${XSPDB};
                          tablename=VAL_ADS2;
                          columns=tot_age,tot_income,tot_cust_years,tot_children,single_in
d,married_ind,separated_ind,female_ind,ck_acct_ind,sv_acct_ind,sv_avg_bal,ck_avg_bal;
                          dependent=cc_acct_ind;
                          outputdatabase=${QLID};
                          outputtablename=LogisticOut2;
                          statstable=true;
                          successtable=true;
                          thresholdtable=true;
                          lifttable=true;
                          groupby=state_code');
```

Out[37]:

```
Success: 0 rows affected
```

In [38]:

```
SELECT * FROM ${QLID}.LogisticOut2 ORDER BY 1,3;
```

Out[38]:

	state_code	Column Name	B Coefficient	
1	AZ	(Constant)	-222.177333380171	27
2	AZ	tot_children	-73.2981397159156	87
3	AZ	sv_acct_ind	-40.9050539756336	72
4	AZ	tot_income	-0.000571943999680645	0.0
5	AZ	ck_avg_bal	0.00428472111672978	0.1
6	AZ	sv_avg_bal	0.0338129422350836	0.4
7	AZ	tot_age	2.06221659283055	26
8	AZ	tot_cust_years	7.53801718492202	87
9	AZ	female_ind	10.8079622892211	76
10	AZ	single_ind	30.9771356695184	13
11	AZ	ck_acct_ind	84.4354483052293	93
12	AZ	married_ind	85.0366927090602	16
13	AZ	separated_ind	260.97461115321	30
14	CA	(Constant)	-3.1844166219965	1.0
15	CA	married_ind	-0.122746392049839	0.5
16	CA	tot_cust_years	-0.0779959834070567	0.0
17	CA	sv_avg_bal	-0.00066353105703381	0.0
18	CA	ck_avg_bal	-0.000180310517872725	0.0
19	CA	tot_income	3.51387623963837e-05	1.1
20	CA	tot_age	0.0326684230020237	0.0
21	CA	tot_children	0.155460504740809	0.1
22	CA	female_ind	0.280781364997123	0.3
23	CA	sv_acct_ind	0.898072321656622	0.4
24	CA	ck_acct_ind	1.39404777277645	0.4
25	CA	single_ind	1.4682871055734	0.7
26	CA	separated_ind	1.85934008412688	1.2
27	IL	tot_children	-0.699550465091909	0.6

In [39]:

```
SELECT * FROM ${QLID}.LogisticOut2_rpt ORDER BY 1;
```

Out[39]:

	rid	state_code	Total Observations	Total Iterations	Initial Log Likelihood	Final Log Likelihood
1	1	AZ	24	14	-15.8775177157916	-0.000216312526332766
2	2	CA	177	6	-120.619667195349	-92.173936126782
3	3	IL	56	10	-37.047541336847	-25.6980241897872
4	4	NY	107	6	-66.1071188619215	-53.5688071505545
5	5	OH	23			
6	6	OTHER	277	5	-180.578512098577	-143.543527623196
7	7	TX	83	5	-56.5089397714094	-45.8826898970036

In [40]:

```
SELECT * FROM ${QLID}.LogisticOut2_txt ORDER BY 1;
```

Out[40]:

	state_code	partId	XmlModel
1	AZ	1	NOS_USRVAL_ADS212cc_acct_ind1none0falseIncludeConstanttot_age47.8
2	CA	1	NOS_USRVAL_ADS212cc_acct_ind1none0falseIncludeConstanttot_age38.8
3	IL	1	NOS_USRVAL_ADS212cc_acct_ind1none0falseIncludeConstanttot_age39.8
4	NY	1	NOS_USRVAL_ADS212cc_acct_ind1none0falseIncludeConstanttot_age44.4
5	OH	1	NOS_USRVAL_ADS212cc_acct_ind1none0falseIncludeConstantConstant co
6	OTHER	1	NOS_USRVAL_ADS212cc_acct_ind1none0falseIncludeConstanttot_age43.7
7	TX	1	NOS_USRVAL_ADS212cc_acct_ind1none0falseIncludeConstanttot_age43.2

In [41]:

```
SELECT XMLSERIALIZE(Content X.Dot) as XMLText
FROM (SELECT * FROM ${QLID}.LogisticOut2_txt) AS C,
XMLTable (
'//*'
PASSING CREATEXML(C.XmlModel)
) AS X ("Dot")
;
```

Out[41]:

	XMLText
1	NOS_USRVAL_ADS212cc_acct_ind1none0falseIncludeConstanttot_age39.8392857142857
2	NOS_USRVAL_ADS212cc_acct_ind1none0falseIncludeConstanttot_age44.46728971962616
3	NOS_USRVAL_ADS212cc_acct_ind1none0falseIncludeConstantConstant columns detected.
4	NOS_USRVAL_ADS212cc_acct_ind1none0falseIncludeConstanttot_age43.24096385542169
5	NOS_USRVAL_ADS212cc_acct_ind1none0falseIncludeConstanttot_age43.75812274368231
6	NOS_USRVAL_ADS212cc_acct_ind1none0falseIncludeConstanttot_age39.8392857142857
7	NOS_USRVAL_ADS212cc_acct_ind1none0falseIncludeConstanttot_age44.46728971962616
8	NOS_USRVAL_ADS212cc_acct_ind1none0falseIncludeConstantConstant columns detected.
9	NOS_USRVAL_ADS212cc_acct_ind1none0falseIncludeConstanttot_age43.24096385542169
10	NOS_USRVAL_ADS212cc_acct_ind1none0falseIncludeConstanttot_age43.75812274368231
11	NOS_USRVAL_ADS212cc_acct_ind1none0falseIncludeConstanttot_age39.8392857142857
12	NOS_USRVAL_ADS212cc_acct_ind1none0falseIncludeConstanttot_age44.46728971962616
13	NOS_USRVAL_ADS212cc_acct_ind1none0falseIncludeConstantConstant columns detected.
14	NOS_USRVAL_ADS212cc_acct_ind1none0falseIncludeConstanttot_age43.24096385542169
15	NOS_USRVAL_ADS212cc_acct_ind1none0falseIncludeConstanttot_age43.75812274368231
16	NOS_USRVAL_ADS212cc_acct_ind1none0falseIncludeConstanttot_age39.8392857142857
17	NOS_USRVAL_ADS212cc_acct_ind1none0falseIncludeConstanttot_age44.46728971962616
18	NOS_USRVAL_ADS212cc_acct_ind1none0falseIncludeConstantConstant columns detected.
19	NOS_USRVAL_ADS212cc_acct_ind1none0falseIncludeConstanttot_age43.24096385542169
20	NOS_USRVAL_ADS212cc_acct_ind1none0falseIncludeConstanttot_age43.75812274368231
21	NOS_USRVAL_ADS212cc_acct_ind1none0falseIncludeConstanttot_age39.8392857142857
22	NOS_USRVAL_ADS212cc_acct_ind1none0falseIncludeConstanttot_age44.46728971962616
23	NOS_USRVAL_ADS212cc_acct_ind1none0falseIncludeConstantConstant columns detected.
24	NOS_USRVAL_ADS212cc_acct_ind1none0falseIncludeConstanttot_age43.24096385542169
25	NOS_USRVAL_ADS212cc_acct_ind1none0falseIncludeConstanttot_age43.75812274368231
26	NOS_USRVAL_ADS212cc_acct_ind1none0falseIncludeConstanttot_age39.8392857142857
27	NOS_USRVAL_ADS212cc_acct_ind1none0falseIncludeConstanttot_age44.46728971962616

Scoring da Regressão Logística

Objetivo

Depois de construir um modelo preditivo usando a função de Regressão Logística, o modelo pode ser passado para uma função de Scoring de Regressão Logística para a criação de uma tabela de scoring contendo os valores previstos da variável dependente. Isso é feito lendo o banco de dados de saída e o nome da tabela de saída criados pela função de regressão logística, referindo-se a eles aqui como banco de dados de modelo e nome de tabela de modelo, respectivamente.

Além de uma tabela de pontuação, esta função pode opcionalmente produzir os seguintes resultados.

- Succes Table
- Multi-Threshold Success Table
- Lift Table

---

1. Primeiro, vamos executar o scoring do modelo singular de regressão logística criado acima. Vamos também criar uma Lift Table e uma Sucess Table, preenchidas em uma tabela com _score_txt ao fim do nome da tabela de modelo.

In [42]:

```
call ${XSPDB}.td_analyze('logisticscore',
                         'database=${XSPDB};
                          tablename=VAL_ADS;
                          modeldatabase=${QLID};
                          modeltablename=LogisticOut1;
                          outputdatabase=${QLID};
                          outputtablename=LogisticScore1;
                          estimate=Estimate;
                          probability=Probability;
                          retain=cc_acct_ind;
                          samplescoresize=10;
                          lifttable=true;
                          successtable=true;
                          scoringmethod=scoreandevaluate');
```

Out[42]:

```
Success: 0 rows affected

[Teradata Database] [Warning 3212] The stored procedure returned one or more
result sets.
```

Out[42]:

	cust_id	cc_acct_ind	Probability	Estimate
1	1362765	1	0.7134597180915243	1
2	1362862	0	0.1618428270762774	0
3	1363232	1	0.40098090847502604	0
4	1363251	0	0.44822035268809973	0
5	1362852	1	0.6283042638327425	1
6	1362596	0	0.27013310963487935	0
7	1363333	0	0.8160701533675125	1
8	1363005	0	0.12947945866983807	0
9	1362526	1	0.8064044536195069	1
10	1363178	0	0.7105647899149466	1

In [43]:

```
SELECT * FROM ${QLID}.LogisticScore1_txt;
```

Out[43]:

	partId	XmlModel
1	1	Threshold Prob___Case 11_____Case 10_____Case 01_____Case 000.0

In [44]:

```
SELECT XMLSERIALIZE(Content X.Dot) as XMLText
FROM (SELECT * FROM ${QLID}.LogisticScore1_txt) AS C,
XMLTable (
'//*'
PASSING CREATEXML(C.XmlModel)
) AS X ("Dot")
;
```

Out[44]:

	XMLText
1	Threshold Prob___Case 11_____Case 10_____Case 01_____Case 000.00__
2	Threshold Prob___Case 11_____Case 10_____Case 01_____Case 000.00__
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	
20	
21	
22	
23	
24	
25	
26	
27	

1. Em seguida, vamos executar o scoring do modelo de regressão logística criado acima para cada estado.

In [45]:

```
call ${XSPDB}.td_analyze('logisticscore',
                         'database=${XSPDB};
                          tablename=VAL_ADS2;
                          modeldatabase=${QLID};
                          modeltablename=LogisticOut2;
                          outputdatabase=${QLID};
                          outputtablename=LogisticScore2;
                          estimate=Estimate;
                          probability=Probability;
                          retain=cc_acct_ind;
                          samplescoresize=10;
                          scoringmethod=score');
```

Out[45]:

```
Success: 0 rows affected

[Teradata Database] [Warning 3212] The stored procedure returned one or more
result sets.
```

Out[45]:

	cust_id	state_code	cc_acct_ind	Probability	Estimate
1	1362677	OTHER	0	0.7532874523610942	1
2	1362883	IL	1	0.39178240100620165	0
3	1362728	NY	0	0.5541210015890233	1
4	1363006	TX	1	0.687730236608376	1
5	1363017	CA	1	0.5811144546908545	1
6	1362960	OTHER	1	0.27557520692375326	0
7	1362602	CA	1	0.9161636297952092	1
8	1362909	IL	1	0.9999999999979262	1
9	1362862	CA	0	0.16574576878800046	0
10	1362836	OTHER	1	0.9088645138147252	1

Power BI

Para finalizar este notebook, vamos visualizar o resultado da regressão logística através do PowerBI para identificar qual variável possui maior influência!