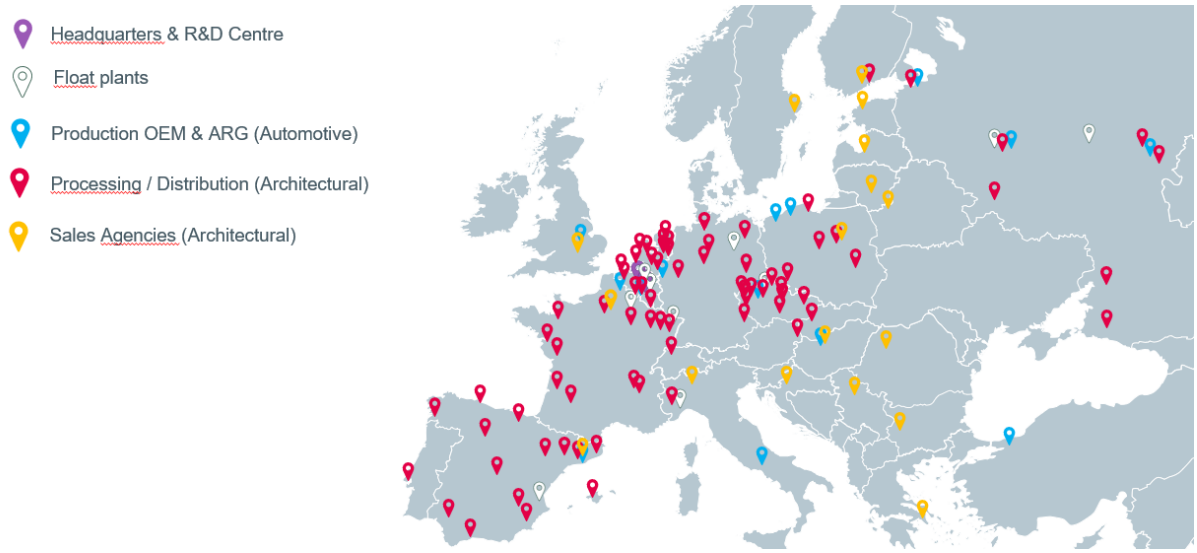
Annexe :

Annexe 1 - Information AGC Glass Europe

Annexe 1.1 – Usines et bâtiments d'AGC en Europe



Source : AGC Glass Europe

Annexe 1.2 – Exemple de liste de prélèvement ou outbound delivery

PICKING LIST

Date: 10.05.2024 **Time:** 09:36:10

Warehouse	Loading Dock	Loading Date	Loading Hour	Shipment Status	Shipment Nr	Transporter	ShipCond	Description
WH Moustier	MOU/LOOSE/PLFHOM/O	13.05.2024	06:00:00			LANNUTTI S.A.	Z1	Inloader truck

Customer	Name	Delivery Nr	Sales Nr	Delivery Block	Total Weight
3100504	SCHOTT Flat Glass CR, s.r.o. Za#ovsk6 850,VALA#SKÁ MEZI#H#H,757 01,CZ	8001218240	7001095100	:	24.549,279 KG

Delivery Line	Material	Stillage Type	Platform	Required Criteria	Stillage Grouping	Pick Qty
000010	3088178	<CHEVALET P10 PLF L-FRAME>		CC		5.000BLK

Material Description FCLL 3.15P1 6000032100V00 18 — **Stillage Mandatory** — **Weight** 13607KG Loading Driver's Side — **Req. Qty** 5.000BLK — **Ins.Flg**

MRN2

Loading Instructions

Loading Instructions <CHEVALET P10 PLF L-FRAME>

Other Suggestions:

Rack/Bin	Stillage	Block ID	Criteria	Criteria Description
X3D11/25		MS01490690		

Delivery Line	Material	Stillage Type	Platform	Required Criteria	Stillage Grouping	Pick Qty
000020	3088735	<CHEVALET P10 PLF L-FRAME>		CC/CI/CI		3.000BLK

Material Description FCLLLM 3.15P1 6000032100V00 18 — **Stillage Mandatory** — **Weight** 8164KG Loading Driver's Side/Coating Side In/Coating Side In — **Req. Qty** 3.000BLK — **Ins.Flg**

Loading Instructions

Loading Instructions <CHEVALET P10 PLF L-FRAME>

Other Suggestions:

Rack/Bin	Stillage	Block ID	Criteria	Criteria Description

Source : AGC Glass Europe

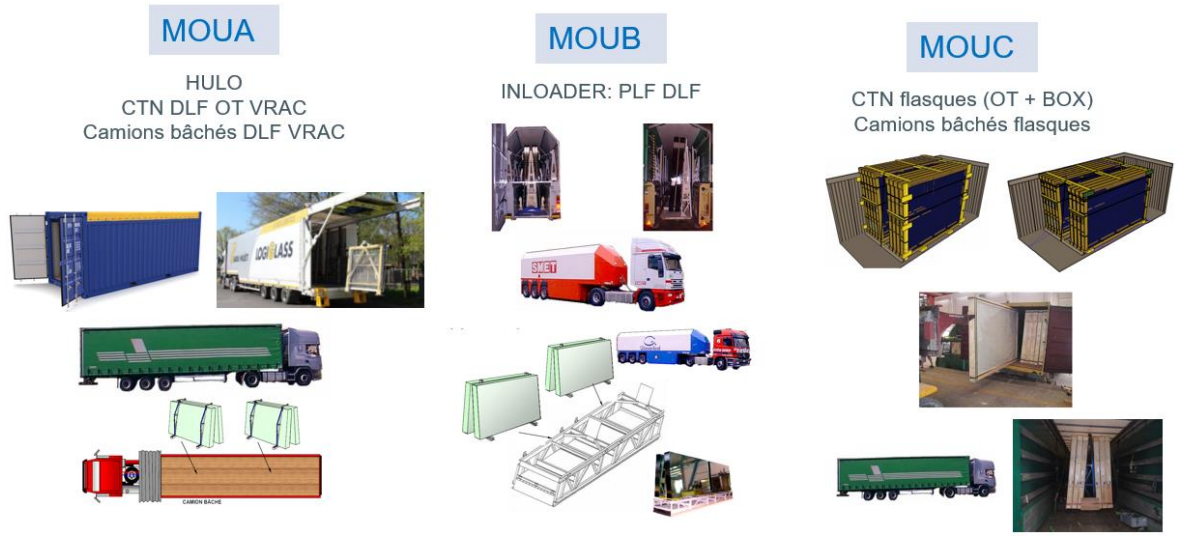
Annexe 1.3 - Transpupitres

Source : https://www.hubtex.com/fr-be/produits/systemes-de-transport-du-verre/transpupitres-pour-verres-plats-gtr

Annexe 1.4 - Chevalet

Source : https://www.rotomshop.fr/blogs/blog/les-chevalets-et-pupitres-porte-verre/

Annexe 1.5 – Différents embarcadères à l'usine de Moustier



Source : AGC Glass Europe

Annexe 1.6 – Carnet de commande AGC

MOUA

Sem	Jour	MOU/LOOSE/DLF/OUT	MOU/LDCSOFTSIDED/OUT	Blocage avec Ressource	F7	DUMMY	<EMPTY>	Total	Capacité
18	29-04-24	9	2					11	
	30-04-24	8						8	
	01-05-24								
	02-05-24	4	4					8	
	03-05-24	2						2	
	04-05-24								
Moyenne 18		5,8						7,25	
19	06-05-24	7						7	
	07-05-24	1	2					3	
	08-05-24	1						1	
	09-05-24								
	10-05-24	4						4	4
	11-05-24								
Moyenne 19		3,3						3,75	4
20	13-05-24	9	3					12	9
	14-05-24	6						6	7
	15-05-24	11	2					13	11
	16-05-24	10	1					11	10
	17-05-24	5						5	7
	18-05-24								
Moyenne 20		8,2						9,4	8,8
21	20-05-24								
	21-05-24	6						6	6
	22-05-24	4						4	4
	23-05-24	4			1			4	5
	24-05-24	7	2					9	16
	25-05-24								
Moyenne 21		5,3						5,75	7,75
22	27-05-24	5						5	5
	28-05-24	5						5	5
	29-05-24	5						5	5
	30-05-24	5						5	5
	31-05-24	6			2			6	8
	01-06-24								
Moyenne 22		5,2						5,2	5,6

MOUB

Jour	MOU/LOOSE/INLDLF/OUT	MOU/LOOSE/INLDLFLAM/OUT	MOU/LOOSE/PLFHOM/OUT	MOU/LOOSE/PLFLAM/OUT	MOU/LOOSE/PLFMIX/OUT	MOU/LOOSE/LDCINLOADER/OUT	Blocage avec Ressource	F7	<EMPTY>	Total	Capacité
29-04-24	7	4	40	23	9					83	
30-04-24	10	6	29	22	10					77	
01-05-24											
02-05-24	6	5	33	24	17			1		85	
03-05-24	9	6	34	14	16					79	
04-05-24											
Moyenne 18	8	5,3	34	21	13					81	
06-05-24	12	7	26	23	19					87	
07-05-24	11	4	21	22	10					68	
08-05-24	6	1	17	10	8		1			42	
09-05-24											
10-05-24	4	6	21	24	10		1			65	65
11-05-24											
Moyenne 19	8,3	4,5	21	20	12					65,5	65
13-05-24	11	7	32	27	15					92	111
14-05-24	9	7	28	18	12					74	105
15-05-24	7	5	32	26	14		1			84	106
16-05-24	7	2	15	21	8		3			53	103
17-05-24	4	5	29	19	8		4			65	103
18-05-24											
Moyenne 20	7,6	5,2	27	22	11					73,6	106
20-05-24											
21-05-24	10	4	39	23	15					91	103
22-05-24	2	2	31	25	3		1			63	105
23-05-24	3		31	10	4		2			48	98
24-05-24	5	3	25	9	4					46	103
25-05-24											
Moyenne 21	5	3	32	17	6,5					62	102
27-05-24	3	4	30	7	4		4			48	103
28-05-24	1	2	15	7	6		5			31	103
29-05-24	1	2	26	6	4		12			39	103
30-05-24	5	1	20	4	6		6			36	103
31-05-24	1	1	31	3	4					40	103
01-06-24											
Moyenne 22	2,2	2	24	5,4	4,8					38,8	103

MOUC

Sem	Jour	MOU/ENDCAPS/SOFT/OUT	MOU/ENDCAPS/CNT-OUT	Blocage avec Ressource	F7	<EMPTY>	DUMMY	Total	Capacité
18	29-04-24	8	15					23	
	30-04-24	3	8					11	
	01-05-24								
	02-05-24	10	16					26	
	03-05-24	10	19					29	
	04-05-24								
Moyenne 18		7,8	15					22,3	
19	06-05-24	9	13					22	
	07-05-24	6	12					18	
	08-05-24	8	13					21	
	09-05-24								
	10-05-24	7	13					20	20
	11-05-24								
Moyenne 19		7,5	13					20,3	20
20	13-05-24	9	27		1			36	36
	14-05-24	10	17		3			27	27
	15-05-24	9	15					24	28
	16-05-24	5	17					22	28
	17-05-24	4	21					25	28
	18-05-24								
Moyenne 20		7,4	19					26,8	29
21	20-05-24								
	21-05-24	8	20		1			28	28
	22-05-24	7	18				3	25	28
	23-05-24	6	17		1			23	28
	24-05-24	8	17				1	25	28
	25-05-24								
Moyenne 21		7,3	18					25,3	28
22	27-05-24	8	19		1			27	28
	28-05-24	4	13					17	28
	29-05-24	5	5					10	28
	30-05-24	6	2					8	28
	31-05-24	4	1					5	28
	01-06-24								
Moyenne 22		5,4						13,4	28

Source : AGC Glass Europe

Annexe 2 – Matrice de confusion

Confusion matrix		Prédite	
		0	1
Vrai	0	Vrai Négatif (TN)	Faux positif (FP)
	1	Faux Négatif (FN)	Vrai Positif (TP)

Pour expliquer le fonctionnement de la matrice de confusion :

Vrai négatif (TN) : Nombre de fois où le modèle a correctement prédit la classe "Non" (négative).
Faux positif (FP) : Nombre de fois où le modèle a prédit "Oui" (positive) alors que la classe réelle était "Non" (négative). Également appelé erreur de Type I.
Faux négatif (FN) : Nombre de fois où le modèle a prédit "Non" (négative) alors que la classe réelle était "Oui" (positive). Également appelé erreur de Type II.
Vrai positif (TP) : Nombre de fois où le modèle a correctement prédit la classe "Oui" (positive).

A partir de ses informations, nous pouvons calculer plusieurs métriques importantes pour évaluer les performances d'un modèle de machine learning :

NOM	Formule	Définition
Accuracy	$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$	Cette valeur indique la proportion de prédictions correctes sur l'ensemble des prédictions.
Precision	$\text{Precision} = \frac{TP}{TP + FP}$	Cette valeur mesure la proportion de prédiction positive correcte parmi toutes les prédictions positive du modèle.
Recall	$\text{Recall} = \frac{TP}{TP + FN}$	Cette valeur représente la proportion de vraies observations positive correctement identifiées par le modèle.
F1 - Score	$F1 - \text{Score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$	Cette métrique représente la moyenne de la précision et du recall offrant un équilibre entre les deux métriques.

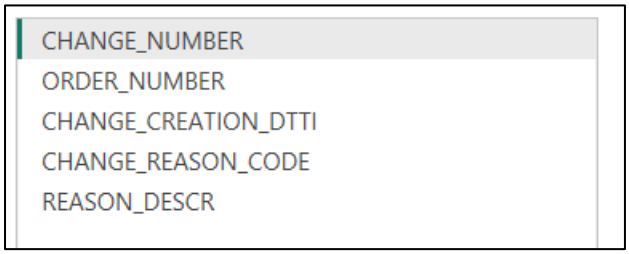
Source : Narkhede, S. (2018). *Understanding confusion matrix*. Towards Data Science. En ligne sur https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62

Annexe 3 – PowerBI

Annexe 3.1 - Table de données initiales

SHIPMENT :

ORDER_MODIF :



Annexe 3.2 – Code M pour la table ORDER_MODIF

```
let
    Source = ORDER_MODIFS,
    #"Extracted Last Characters" = Table.TransformColumns(Source, {{"CHANGE_REASON_CODE", each Text.End(_, 1), type text}}),
    #"Added Conditional Column" = Table.AddColumn(#"Extracted Last Characters", "MODIF_DUE_CUST", each if [CHANGE_REASON_CODE] = "2" then 1 else if [CHANGE_REASON_CODE] = "6" then 1 else 0),
```

```
    #"Added Conditional Column1" = Table.AddColumn(#"Added Conditional Column", "MODIF_DUE_AGC", each if [CHANGE_REASON_CODE] = "4" then 1 else if [CHANGE_REASON_CODE] = "7" then 1 else 0),
    #"Grouped Rows" = Table.Group(#"Added Conditional Column1", {"ORDER_NUMBER"}, {{"MODIF_LAST_DTTI", each List.Max([CHANGE_CREATION_DTTI]), type nullable datetime}, {"MODIF_NUMBER", each Table.RowCount(_), Int64.Type}, {"MODIF_DUE_CUST", each List.Max([MODIF_DUE_CUST]), type number}, {"MODIF_DUE_AGC", each List.Max([MODIF_DUE_AGC]), type number}})
in
    #"Grouped Rows"
```

Annexe 3.3 – Code M pour la table SHIPMENT

```
let
Source = #"SHIPMENT (B)",
    #"Removed Other Columns" = Table.SelectColumns(Source,{"ORDER_NB", "ORDER_DT", "CUSTOMER_SHIP_TO_CODE", "FACTORY_CODE", "SHIPMENT_NB", "SUSPEND_CODE", "PROJECT_CODE", "DELIVERY_NB", "SHIPPING_POINT_CODE", "SHIPMENT_CONDITION_DESCR", "REAL_SHIP_COMP_DATE"}),
    #"Filtered Rows" = Table.SelectRows(#"Removed Other Columns", each ([DELIVERY_NB] <> "") and ([REAL_SHIP_COMP_DATE] <> null)),
    #"Added Conditional Column" = Table.AddColumn(#"Filtered Rows", "SHIP_SUSPEND_CODE", each if [SUSPEND_CODE] = "F4" then 1 else 0),
    #"Added Conditional Column1" = Table.AddColumn(#"Added Conditional Column", "SHIP_PROJECT_CODE", each if [PROJECT_CODE] = "" then 0 else 1),
    #"Grouped Rows" = Table.Group(#"Added Conditional Column1", {"DELIVERY_NB", "ORDER_NB", "CUSTOMER_SHIP_TO_CODE", "FACTORY_CODE", "SHIPPING_POINT_CODE", "SHIPMENT_CONDITION_DESCR", "SHIPMENT_NB"}, {{"SHIP_ORDER_DTTI", each List.Max([ORDER_DT]), type nullable datetime}, {"SHIP_LAST_DTTI", each List.Max([REAL_SHIP_COMP_DATE]), type nullable datetime}, {"SHIP_SUSPEND_CODE", each List.Max([SHIP_SUSPEND_CODE]), type number}, {"SHIP_PROJECT_CODE", each List.Max([SHIP_PROJECT_CODE]), type number}}),
    #"Filtered Rows1" = Table.SelectRows(#"Grouped Rows", each ([SHIPMENT_CONDITION_DESCR] <> "AGC Cust Truck" and [SHIPMENT_CONDITION_DESCR] <> "AGC Distrib Truck" and [SHIPMENT_CONDITION_DESCR] <> "AGC Plane" and [SHIPMENT_CONDITION_DESCR] <> "AGC Small Platform")),
```

```
    #"Filtered Rows2" = Table.SelectRows(#"Filtered Rows1", each [DELIVERY_NB] <>
"8001074173" and [DELIVERY_NB] <> "8001074583")
in
    #"Filtered Rows2"
```

Annexe 3.4 – Code M pour la table CUSTOMERS

```
let
    Source = #"SHIPMENT (B)",
    #"Removed Other Columns" = Table.SelectColumns(Source,{"ORDER_NB", "ORDER_DT",
"ITEM_NB", "DELIVERY_NB", "CUST_NAME1", "CUST_CLUSTER_DESCR",
"CUST_COUNTRY_NAME", "CUST_TYPE_DESCR", "CUST_CATEGORY_DESCR",
"CUST_ACTIVITY_DESCR", "CUST_GROUP_DESCR"}),
    #"Reordered Columns" = Table.ReorderColumns(#"Removed Other
Columns",{"DELIVERY_NB", "ORDER_NB", "ORDER_DT", "ITEM_NB", "CUST_NAME1",
"CUST_CLUSTER_DESCR", "CUST_COUNTRY_NAME", "CUST_TYPE_DESCR",
"CUST_CATEGORY_DESCR", "CUST_ACTIVITY_DESCR", "CUST_GROUP_DESCR"}),
    #"Filtered Rows" = Table.SelectRows(#"Reordered Columns", each ([DELIVERY_NB] <> "")),
    #"Grouped Rows" = Table.Group(#"Filtered Rows", {"DELIVERY_NB", "ORDER_NB",
"CUST_NAME1", "CUST_CLUSTER_DESCR", "CUST_COUNTRY_NAME",
"CUST_TYPE_DESCR", "CUST_CATEGORY_DESCR", "CUST_ACTIVITY_DESCR",
"CUST_GROUP_DESCR"}, {{"CUST_ORDER_DT", each List.Max([ORDER_DT]), type nullable
datetime}})
in
    #"Grouped Rows"
```

Annexe 3.5 – Code M pour la table MATERIALS

```
let
      Source = #"SHIPMENT (B)",
    #"Filtered Rows2" = Table.SelectRows(Source, each not Text.StartsWith([ORDER_NB], "4")),
    #"Removed Other Columns" = Table.SelectColumns(#"Filtered
Rows2",{"DELIVERY_NB","MATERIAL_CODE", "MAT_PODGL_NPH_BU_CODE",
"MAT_DIMENSION_TYPE",   "MAT_PACKAGING_TYPE_DESCR",
"SUPPLY_CHAIN_LEAD_TIME_CODE", "PLATFORM", "STILLAGE_TYPE",
"MIXED_ORDER_BL", "MIXED_BLOCK_BL", "MIXED_END_CAP_BL"}),
    #"Filtered Rows" = Table.SelectRows(#"Removed Other Columns", each ([DELIVERY_NB] <>
"")),
```

```
    #"Replaced Value" = Table.ReplaceValue(#"Filtered
Rows","OR","NLT",Replacer.ReplaceText,{"SUPPLY_CHAIN_LEAD_TIME_CODE"}),
    #"Added Conditional Column" = Table.AddColumn(#"Replaced Value", "MAT_BU_FLOAT",
each if [MAT_PODGL_NPH_BU_CODE] = "20" then 1 else 0),
    #"Added Conditional Column1" = Table.AddColumn(#"Added Conditional Column",
"MAT_BU_COATING", each if [MAT_PODGL_NPH_BU_CODE] = "21" then 1 else 0),
    #"Added Conditional Column2" = Table.AddColumn(#"Added Conditional Column1",
"MAT_BU_LAM", each if [MAT_PODGL_NPH_BU_CODE] = "22" then 1 else 0),
    #"Added Conditional Column3" = Table.AddColumn(#"Added Conditional Column2",
"MAT_BU_DECO", each if [MAT_PODGL_NPH_BU_CODE] = "23" then 1 else 0),
    #"Added Conditional Column4" = Table.AddColumn(#"Added Conditional Column3",
"MAT_BU_INB", each if [MAT_PODGL_NPH_BU_CODE] = "24" then 1 else 0),
    #"Added Conditional Column5" = Table.AddColumn(#"Added Conditional Column4",
"MAT_BU_CAST", each if [MAT_PODGL_NPH_BU_CODE] = "25" then 1 else 0),
    #"Added Conditional Column6" = Table.AddColumn(#"Added Conditional Column5",
"MAT_DIM_PLF", each if [MAT_DIMENSION_TYPE] = "PLF" then 1 else 0),
    #"Added Conditional Column7" = Table.AddColumn(#"Added Conditional Column6",
"MAT_DIM_DLF", each if [MAT_DIMENSION_TYPE] = "DLF" then 1 else 0),
    #"Added Conditional Column8" = Table.AddColumn(#"Added Conditional Column7",
"MAT_DIM_MF", each if [MAT_DIMENSION_TYPE] = "MF" then 1 else 0),
    #"Added Conditional Column9" = Table.AddColumn(#"Added Conditional Column8",
"MAT_PACK_BOXE", each if [MAT_PACKAGING_TYPE_DESCR] = "Boxes" then 1 else 0),
    #"Added Conditional Column10" = Table.AddColumn(#"Added Conditional Column9",
"MAT_PACK_LOOSE", each if [MAT_PACKAGING_TYPE_DESCR] = "Loose" then 1 else 0),
    #"Added Conditional Column11" = Table.AddColumn(#"Added Conditional Column10",
"MAT_PACK_ENDCAP", each if [MAT_PACKAGING_TYPE_DESCR] = "Endcaps" then 1 else
0),
    #"Added Conditional Column12" = Table.AddColumn(#"Added Conditional Column11",
"MAT_LT_STX", each if [SUPPLY_CHAIN_LEAD_TIME_CODE] = "STX" then 1 else 0),
    #"Added Conditional Column13" = Table.AddColumn(#"Added Conditional Column12",
"MAT_LT_ST1", each if [SUPPLY_CHAIN_LEAD_TIME_CODE] = "ST1" then 1 else 0),
    #"Added Conditional Column14" = Table.AddColumn(#"Added Conditional Column13",
"MAT_LT_NLT", each if [SUPPLY_CHAIN_LEAD_TIME_CODE] = "NLT" then 1 else 0),
    #"Added Conditional Column15" = Table.AddColumn(#"Added Conditional Column14",
"MAT_PLATFORM", each if [PLATFORM] = "" then "No" else "Yes"),
    #"Inserted First Characters" = Table.AddColumn(#"Added Conditional Column15", "First
Characters", each Text.Start([STILLAGE_TYPE], 1), type text),
```

```
    #"Added Conditional Column16" = Table.AddColumn(#"Inserted First Characters",
"MAT_STILLAGE", each if [First Characters] = "P" then "PLF Stillage" else if [First Characters] =
"D" then "DLF Stillage" else "Autres"),
    #"Grouped Rows" = Table.Group(#"Added Conditional Column16", {"DELIVERY_NB",
"MAT_PLATFORM"}, {{"MAT_NUMBER", each Table.RowCount(_), Int64.Type},
{"MAT_BU_FLOAT", each List.Max([MAT_BU_FLOAT]), type number},
{"MAT_BU_COATING", each List.Max([MAT_BU_COATING]), type number},
{"MAT_BU_LAM", each List.Max([MAT_BU_LAM]), type number}, {"MAT_BU_DECO", each
List.Max([MAT_BU_DECO]), type number}, {"MAT_BU_IND", each List.Max([MAT_BU_INB]),
type number}, {"MAT_BU_CAST", each List.Max([MAT_BU_CAST]), type number},
{"MAT_DIM_PLF", each List.Max([MAT_DIM_PLF]), type number}, {"MAT_DIM_DLF", each
List.Max([MAT_DIM_DLF]), type number}, {"MAT_DIM_MF", each List.Max([MAT_DIM_MF]),
type number}, {"MAT_PACK_BOXE", each List.Max([MAT_PACK_BOXE]), type number},
{"MAT_PACK_LOOSE", each List.Max([MAT_PACK_LOOSE]), type number},
{"MAT_PACK_ENDCAP", each List.Max([MAT_PACK_ENDCAP]), type number},
{"MAT_LT_STX", each List.Max([MAT_LT_STX]), type number}, {"MAT_LT_ST1", each
List.Max([MAT_LT_ST1]), type number}, {"MAT_LT_NLT", each List.Max([MAT_LT_NLT]),
type number}, {"MAT_STILLAGE", each List.Max([MAT_STILLAGE]), type text},
{"MAT_MIXED_ORDER_BL", each List.Max([MIXED_ORDER_BL]), type nullable number},
{"MAT_MIXED_BLOCK_BL", each List.Max([MIXED_BLOCK_BL]), type nullable number},
{"MAT_MIXED_END_CAP_BL", each List.Max([MIXED_END_CAP_BL]), type nullable
number}}),
    #"Added Custom" = Table.AddColumn(#"Grouped Rows", "MAT_BU_TOTAL", each
[MAT_BU_FLOAT]+[MAT_BU_COATING]+[MAT_BU_LAM]+[MAT_BU_DECO]+[MAT_BU_
IND]+[MAT_BU_CAST]),
    #"Reordered Columns" = Table.ReorderColumns(#"Added Custom",{"DELIVERY_NB",
"MAT_PLATFORM", "MAT_STILLAGE", "MAT_MIXED_ORDER_BL",
"MAT_MIXED_BLOCK_BL", "MAT_MIXED_END_CAP_BL", "MAT_NUMBER",
"MAT_BU_FLOAT", "MAT_BU_COATING", "MAT_BU_LAM", "MAT_BU_DECO",
"MAT_BU_IND", "MAT_BU_CAST", "MAT_BU_TOTAL", "MAT_DIM_PLF",
"MAT_DIM_DLF", "MAT_DIM_MF", "MAT_PACK_BOXE", "MAT_PACK_LOOSE",
"MAT_PACK_ENDCAP", "MAT_LT_STX", "MAT_LT_ST1", "MAT_LT_NLT"})
in
    #"Reordered Columns"
```

Annexe 3.6 – Code M pour la table COMBILOADS

```
let
    Source = PowerPlatform.Dataflows(null),
    Workspaces = Source{[Id="Workspaces"]}[Data],
    #"d6c35fb6-bf9c-47c1-90a5-0bb583239939" = Workspaces{[workspaceId="d6c35fb6-bf9c-47c1-90a5-0bb583239939"]}[Data],
    #"17d39470-5caa-4bb5-a0f7-c1aef923c128" = #"d6c35fb6-bf9c-47c1-90a5-0bb583239939"{[dataflowId="17d39470-5caa-4bb5-a0f7-c1aef923c128"]}[Data],
    DIM_COMBILOADS_ = #"17d39470-5caa-4bb5-a0f7-c1aef923c128"{[entity="DIM_COMBILOADS",version=""]}[Data],
    #"Filtered Rows1" = Table.SelectRows(DIM_COMBILOADS_, each [LASTPICKINGDATE] > #datetime(2023, 1, 1, 0, 0, 0)),
    #"Removed Other Columns" = Table.SelectColumns(#"Filtered Rows1",{"SHIPMENT_NB", "Type_of_flow"}),
    #"Added Conditional Column" = Table.AddColumn(#"Removed Other Columns", "FLOW_FTL", each if [Type_of_flow] = "FTL" then 1 else 0),
    #"Added Conditional Column1" = Table.AddColumn(#"Added Conditional Column", "FLOW_CBL", each if [Type_of_flow] = "CBL" then 1 else 0),
    #"Added Conditional Column2" = Table.AddColumn(#"Added Conditional Column1", "FLOW_CBLU", each if [Type_of_flow] = "CBLU" then 1 else 0),
    #"Added Conditional Column3" = Table.AddColumn(#"Added Conditional Column2", "FLOW_CBU", each if [Type_of_flow] = "CBU" then 1 else 0),
    #"Added Conditional Column4" = Table.AddColumn(#"Added Conditional Column3", "FLOW_GROUPAGE", each if [Type_of_flow] = "GROUPAGE" then 1 else 0),
    #"Grouped Rows" = Table.Group(#"Added Conditional Column4", {"SHIPMENT_NB"}, {{"FLOW_FTL", each List.Max([FLOW_FTL]), type number}, {"FLOW_CBL", each List.Max([FLOW_CBL]), type number}, {"FLOW_CBLU", each List.Max([FLOW_CBU]), type number}, {"FLOW_CBU", each List.Max([FLOW_CBU]), type number}, {"FLOW_GROUPAGE", each List.Max([FLOW_GROUPAGE]), type number}}),
    #"Changed Type" = Table.TransformColumnTypes(#"Grouped Rows",{{"FLOW_FTL", Int64.Type}, {"FLOW_CBL", Int64.Type}, {"FLOW_CBLU", Int64.Type}, {"FLOW_CBU", Int64.Type}, {"FLOW_GROUPAGE", Int64.Type}})
in
    #"Changed Type"
```

Annexe 3.7 – Code M pour la table FINALTABLE

```
let
    Source = Table.NestedJoin(SHIPPING, {"ORDER_NB"}, #"ORDER_MODIFS (SO)",
{"ORDER_NUMBER"}, "ORDER_MODIFS (SO)", JoinKind.LeftOuter),
    #"Expanded ORDER_MODIFS (SO)" = Table.ExpandTableColumn(Source, "ORDER_MODIFS
(SO)", {"MODIF_LAST_DTTI", "MODIF_NUMBER", "MODIF_DUE_CUST",
"MODIF_DUE_AGC"}, {"ORDER_MODIFS (SO).MODIF_LAST_DTTI", "ORDER_MODIFS
(SO).MODIF_NUMBER", "ORDER_MODIFS (SO).MODIF_DUE_CUST", "ORDER_MODIFS
(SO).MODIF_DUE_AGC"}),
    #"Replaced Value" = Table.ReplaceValue(#"Expanded ORDER_MODIFS
(SO)",null,0,Replacer.ReplaceValue,{"ORDER_MODIFS (SO).MODIF_NUMBER",
"ORDER_MODIFS (SO).MODIF_DUE_CUST", "ORDER_MODIFS (SO).MODIF_DUE_AGC"}),
    #"Added Conditional Column" = Table.AddColumn(#"Replaced Value", "MODIF_LAST_DTTI",
each if [#"ORDER_MODIFS (SO).MODIF_LAST_DTTI"] = null then [SHIP_ORDER_DTTI] else
[#"ORDER_MODIFS (SO).MODIF_LAST_DTTI"]),
    #"Merged Queries" = Table.NestedJoin(#"Added Conditional Column", {"DELIVERY_NB"},
MATERIALS, {"DELIVERY_NB"}, "MATERIALS", JoinKind.LeftOuter),
    #"Expanded MATERIALS" = Table.ExpandTableColumn(#"Merged Queries", "MATERIALS",
{"MAT_PLATFORM", "MAT_STILLAGE", "MAT_MIXED_ORDER_BL",
"MAT_MIXED_BLOCK_BL", "MAT_MIXED_END_CAP_BL", "MAT_NUMBER",
"MAT_BU_FLOAT", "MAT_BU_COATING", "MAT_BU_LAM", "MAT_BU_DECO",
"MAT_BU_IND", "MAT_BU_CAST", "MAT_BU_TOTAL", "MAT_DIM_PLF",
"MAT_DIM_DLF", "MAT_DIM_MF", "MAT_PACK_BOXE", "MAT_PACK_LOOSE",
"MAT_PACK_ENDCAP", "MAT_LT_STX", "MAT_LT_ST1", "MAT_LT_NLT"},
{"MATERIALS.MAT_PLATFORM", "MATERIALS.MAT_STILLAGE",
"MATERIALS.MAT_MIXED_ORDER_BL", "MATERIALS.MAT_MIXED_BLOCK_BL",
"MATERIALS.MAT_MIXED_END_CAP_BL", "MATERIALS.MAT_NUMBER",
"MATERIALS.MAT_BU_FLOAT", "MATERIALS.MAT_BU_COATING",
"MATERIALS.MAT_BU_LAM", "MATERIALS.MAT_BU_DECO",
"MATERIALS.MAT_BU_IND", "MATERIALS.MAT_BU_CAST",
"MATERIALS.MAT_BU_TOTAL", "MATERIALS.MAT_DIM_PLF",
"MATERIALS.MAT_DIM_DLF", "MATERIALS.MAT_DIM_MF",
"MATERIALS.MAT_PACK_BOXE", "MATERIALS.MAT_PACK_LOOSE",
"MATERIALS.MAT_PACK_ENDCAP", "MATERIALS.MAT_LT_STX",
"MATERIALS.MAT_LT_ST1", "MATERIALS.MAT_LT_NLT"}),
```

```
    #"Merged Queries1" = Table.NestedJoin(#"Expanded MATERIALS", {"DELIVERY_NB"},
CUSTOMERS, {"DELIVERY_NB"}, "CUSTOMERS", JoinKind.LeftOuter),
    #"Expanded CUSTOMERS" = Table.ExpandTableColumn(#"Merged Queries1", "CUSTOMERS",
{"CUST_NAME1", "CUST_CLUSTER_DESCR", "CUST_COUNTRY_NAME",
"CUST_TYPE_DESCR", "CUST_CATEGORY_DESCR", "CUST_ACTIVITY_DESCR",
"CUST_GROUP_DESCR"}, {"CUSTOMERS.CUST_NAME1",
"CUSTOMERS.CUST_CLUSTER_DESCR", "CUSTOMERS.CUST_COUNTRY_NAME",
"CUSTOMERS.CUST_TYPE_DESCR", "CUSTOMERS.CUST_CATEGORY_DESCR",
"CUSTOMERS.CUST_ACTIVITY_DESCR", "CUSTOMERS.CUST_GROUP_DESCR"}),
    #"Merged Queries2" = Table.NestedJoin(#"Expanded CUSTOMERS", {"SHIPMENT_NB"},
COMBILOADS, {"SHIPMENT_NB"}, "COMBILOADS", JoinKind.LeftOuter),
    #"Expanded COMBILOADS" = Table.ExpandTableColumn(#"Merged Queries2",
"COMBILOADS", {"SHIPMENT_NB", "FLOW_FTL", "FLOW_CBL", "FLOW_CBLU",
"FLOW_CBU", "FLOW_GROUPAGE"}, {"COMBILOADS.SHIPMENT_NB",
"COMBILOADS.FLOW_FTL", "COMBILOADS.FLOW_CBL", "COMBILOADS.FLOW_CBLU",
"COMBILOADS.FLOW_CBU", "COMBILOADS.FLOW_GROUPAGE"}),
    #"Filtered Rows" = Table.SelectRows(#"Expanded COMBILOADS", each not
Text.StartsWith([ORDER_NB], "4")),
    #"Filtered Rows3" = Table.SelectRows(#"Filtered Rows", each [DELIVERY_NB] <>
"8001192386" and [DELIVERY_NB] <> "8001112151" and [DELIVERY_NB] <> "8001112122"
and [DELIVERY_NB] <> "8001090092" and [DELIVERY_NB] <> "8001105896" and
[DELIVERY_NB] <> "8001105379"),
    #"Added Conditional Column1" = Table.AddColumn(#"Filtered Rows3", "MODIF_OR_NO", each
if [#"ORDER_MODIFS (SO).MODIF_DUE_CUST"] = 1 then 1 else if [#"ORDER_MODIFS
(SO).MODIF_DUE_AGC"] = 1 then 1 else 0),
    #"Added Conditional Column2" = Table.AddColumn(#"Added Conditional Column1",
"FACTORY_CLUSTER", each if [FACTORY_CODE] = "MOU1" then "North West " else if
[FACTORY_CODE] = "RET1" then "Central CZ-SK-PL-UA-MD" else if [FACTORY_CODE] =
"CUN1" then "Italy" else if [FACTORY_CODE] = "SEI1" then "North West" else if
[FACTORY_CODE] = "BE01" then "D-DACH" else if [FACTORY_CODE] = "OST1" then "D-
DACH" else if [FACTORY_CODE] = "SAG1" then "Iberica" else if [FACTORY_CODE] = "SEI1"
then "North West" else if [FACTORY_CODE] = "ZEE1" then "North West" else if
[FACTORY_CODE] = "KRY1" then "Central CZ-SK-PL-UA-MD" else if [FACTORY_CODE] =
"BAR1" then "Central CZ-SK-PL-UA-MD" else if [FACTORY_CODE] = "LOD1" then "North
West" else if [FACTORY_CODE] = "BOU1" then "North West" else 0),
```

```
    #"Changed Type1" = Table.TransformColumnTypes(#"Added Conditional
Column2",{{"SHIP_ORDER_DTTI", type date}, {"SHIP_LAST_DTTI", type date},
{"MODIF_LAST_DTTI", type date}}),
    #"Added Custom6" = Table.AddColumn(#"Changed Type1", "LEAD_TIME", each
WorkingDay([SHIP_ORDER_DTTI],[SHIP_LAST_DTTI])),
    #"Added Custom" = Table.AddColumn(#"Added Custom6", "DIFF_SHIP_MODIF", each
WorkingDay([MODIF_LAST_DTTI],[SHIP_LAST_DTTI])),
    #"Added Custom2" = Table.AddColumn(#"Added Custom", "BIN_MODIF_2DAY", each if
[DIFF_SHIP_MODIF] < 4 then 1 else 0),
    #"Added Custom3" = Table.AddColumn(#"Added Custom2", "MODIF_2DAY_EX", each
[BIN_MODIF_2DAY]+[#"ORDER_MODIFS (SO).MODIF_DUE_CUST"]),
    #"Added Conditional Column3" = Table.AddColumn(#"Added Custom3",
"MODIF_WITHIN_2DAY", each if [MODIF_2DAY_EX] = 2 then 1 else 0),
    #"Duplicated Column" = Table.DuplicateColumn(#"Added Conditional Column3",
"MODIF_LAST_DTTI", "MODIF_LAST_DTTI - Copy"),
    #"Duplicated Column2" = Table.DuplicateColumn(#"Duplicated Column",
"MODIF_LAST_DTTI", "MODIF_LAST_DTTI - Copy.1"),
    #"Calculated Quarter" = Table.TransformColumns(#"Duplicated
Column2",{{"MODIF_LAST_DTTI - Copy.1", Date.QuarterOfYear, Int64.Type}}),
    #"Duplicated Column1" = Table.DuplicateColumn(#"Calculated Quarter", "MODIF_LAST_DTTI",
"MODIF_LAST_DTTI - Copy.2"),
    #"Extracted Month" = Table.TransformColumns(#"Duplicated Column1",{{"MODIF_LAST_DTTI
- Copy", Date.Month, Int64.Type}}),
    #"Calculated Week of Month" = Table.TransformColumns(#"Extracted
Month",{{"MODIF_LAST_DTTI - Copy.2", Date.WeekOfMonth, Int64.Type}}),
    #"Duplicated Column3" = Table.DuplicateColumn(#"Calculated Week of Month",
"MODIF_LAST_DTTI", "MODIF_LAST_DTTI - Copy.3"),
    #"Calculated Day of Week" = Table.TransformColumns(#"Duplicated
Column3",{{"MODIF_LAST_DTTI - Copy.3", Date.DayOfWeek, Int64.Type}}),
    #"Added Custom4" = Table.AddColumn(#"Calculated Day of Week", "MODIF_DAY", each
Date.DayOfWeekName([MODIF_LAST_DTTI])),
    #"Added Custom5" = Table.AddColumn(#"Added Custom4", "MIX_LeadTime", each
[MATERIALS.MAT_LT_STX]+[MATERIALS.MAT_LT_ST1]+[MATERIALS.MAT_LT_NLT]),
    #"Added Conditional Column4" = Table.AddColumn(#"Added Custom5", "MIX_LT", each if
[MIX_LeadTime] > 1 then 1 else 0),
    #"Added Conditional Column5" = Table.AddColumn(#"Added Conditional Column4",
"MAT_PLATFORM", each if [MATERIALS.MAT_PLATFORM] = "Yes" then 1 else 0),
```

```
    #"Removed Other Columns" = Table.SelectColumns(#"Added Conditional
Column5",{"DELIVERY_NB", "FACTORY_CODE", "SHIPPING_POINT_CODE",
"SHIPMENT_CONDITION_DESCR", "SHIP_ORDER_DTTI", "SHIP_LAST_DTTI",
"SHIP_SUSPEND_CODE", "SHIP_PROJECT_CODE", "ORDER_MODIFS
(SO).MODIF_NUMBER", "ORDER_MODIFS (SO).MODIF_DUE_CUST", "ORDER_MODIFS
(SO).MODIF_DUE_AGC", "MODIF_LAST_DTTI", "MATERIALS.MAT_STILLAGE",
"MATERIALS.MAT_MIXED_ORDER_BL", "MATERIALS.MAT_MIXED_BLOCK_BL",
"MATERIALS.MAT_MIXED_END_CAP_BL", "MATERIALS.MAT_NUMBER",
"MATERIALS.MAT_BU_FLOAT", "MATERIALS.MAT_BU_COATING",
"MATERIALS.MAT_BU_LAM", "MATERIALS.MAT_BU_DECO",
"MATERIALS.MAT_BU_IND", "MATERIALS.MAT_BU_CAST",
"MATERIALS.MAT_BU_TOTAL", "MATERIALS.MAT_DIM_PLF",
"MATERIALS.MAT_DIM_DLF", "MATERIALS.MAT_DIM_MF",
"MATERIALS.MAT_PACK_BOXE", "MATERIALS.MAT_PACK_LOOSE",
"MATERIALS.MAT_PACK_ENDCAP", "MATERIALS.MAT_LT_STX",
"MATERIALS.MAT_LT_ST1", "MATERIALS.MAT_LT_NLT", "CUSTOMERS.CUST_NAME1",
"CUSTOMERS.CUST_CLUSTER_DESCR", "CUSTOMERS.CUST_COUNTRY_NAME",
"CUSTOMERS.CUST_TYPE_DESCR", "CUSTOMERS.CUST_GROUP_DESCR",
"COMBILOADS.FLOW_FTL", "COMBILOADS.FLOW_CBL", "COMBILOADS.FLOW_CBLU",
"COMBILOADS.FLOW_CBU", "COMBILOADS.FLOW_GROUPAGE", "MODIF_OR_NO",
"FACTORY_CLUSTER", "LEAD_TIME", "MODIF_WITHIN_2DAY", "MODIF_LAST_DTTI -
Copy", "MODIF_LAST_DTTI - Copy.1", "MODIF_LAST_DTTI - Copy.2", "MODIF_LAST_DTTI -
Copy.3", "MIX_LT", "MAT_PLATFORM"}),
    #"Renamed Columns" = Table.RenameColumns(#"Removed Other
Columns",{{"ORDER_MODIFS (SO).MODIF_NUMBER", "MODIF_NUMBER"},
{"ORDER_MODIFS (SO).MODIF_DUE_CUST", "MODIF_DUE_CUST"}, {"ORDER_MODIFS
(SO).MODIF_DUE_AGC", "MODIF_DUE_AGC"}, {"MATERIALS.MAT_STILLAGE",
"MAT_STILLAGE"}, {"MATERIALS.MAT_MIXED_ORDER_BL",
"MAT_MIXED_ORDER_BL"}, {"MATERIALS.MAT_MIXED_BLOCK_BL",
"MAT_MIXED_BLOCK_BL"}, {"MATERIALS.MAT_MIXED_END_CAP_BL",
"MAT_MIXED_END_CAP_BL"}, {"MATERIALS.MAT_NUMBER", "MAT_NUMBER"},
{"MATERIALS.MAT_BU_FLOAT", "MAT_BU_FLOAT"},
{"MATERIALS.MAT_BU_COATING", "MAT_BU_COATING"},
{"MATERIALS.MAT_BU_LAM", "MAT_BU_LAM"}, {"MATERIALS.MAT_BU_DECO",
"MAT_BU_DECO"}, {"MATERIALS.MAT_BU_IND", "MAT_BU_IND"},
{"MATERIALS.MAT_BU_CAST", "MAT_BU_CAST"}, {"MATERIALS.MAT_BU_TOTAL",
"MAT_BU_TOTAL"}, {"MATERIALS.MAT_DIM_PLF", "MAT_DIM_PLF"},
```

{"MATERIALS.MAT_DIM_DLF", "MAT_DIM_DLF"}, {"MATERIALS.MAT_DIM_MF", "MAT_DIM_MF"}, {"MATERIALS.MAT_PACK_BOXE", "MAT_PACK_BOXE"}, {"MATERIALS.MAT_PACK_LOOSE", "MAT_PACK_LOOSE"}, {"MATERIALS.MAT_PACK_ENDCAP", "MAT_PACK_ENDCAP"}, {"MATERIALS.MAT_LT_STX", "MAT_LT_STX"}, {"MATERIALS.MAT_LT_ST1", "MAT_LT_ST1"}, {"MATERIALS.MAT_LT_NLT", "MAT_LT_NLT"}, {"CUSTOMERS.CUST_NAME1", "CUST_NAME1"}, {"CUSTOMERS.CUST_CLUSTER_DESCR", "CUST_CLUSTER_DESCR"}, {"CUSTOMERS.CUST_COUNTRY_NAME", "CUST_COUNTRY_NAME"}, {"CUSTOMERS.CUST_TYPE_DESCR", "CUST_TYPE_DESCR"}, {"CUSTOMERS.CUST_GROUP_DESCR", "CUST_GROUP_DESCR"}, {"MODIF_LAST_DTTI - Copy", "MODIF_MONTH"}, {"MODIF_LAST_DTTI - Copy.1", "MODIF_QUARTER"}, {"COMBILOADS.FLOW_FTL", "FLOW_FTL"}, {"COMBILOADS.FLOW_CBL", "FLOW_CBL"}, {"COMBILOADS.FLOW_CBLU", "FLOW_CBLU"}, {"COMBILOADS.FLOW_CBU", "FLOW_CBU"}, {"COMBILOADS.FLOW_GROUPAGE", "FLOW_GROUPAGE"}, {"MODIF_LAST_DTTI - Copy.2", "MODIF_WEEK"}, {"MODIF_LAST_DTTI - Copy.3", "MODIF_DAY"}}),
    #"Changed Type" = Table.TransformColumnTypes(#"Renamed Columns",{{"MAT_BU_TOTAL", Int64.Type}, {"MODIF_OR_NO", Int64.Type}, {"MODIF_WITHIN_2DAY", Int64.Type}, {"MODIF_MONTH", Int64.Type}, {"MODIF_WEEK", Int64.Type}, {"SHIP_SUSPEND_CODE", Int64.Type}, {"SHIP_PROJECT_CODE", Int64.Type}, {"MODIF_NUMBER", Int64.Type}, {"MODIF_DUE_CUST", Int64.Type}, {"MODIF_DUE_AGC", Int64.Type}, {"MAT_MIXED_ORDER_BL", Int64.Type}, {"MAT_MIXED_BLOCK_BL", Int64.Type}, {"MAT_MIXED_END_CAP_BL", Int64.Type}, {"MAT_BU_FLOAT", Int64.Type}, {"MAT_BU_COATING", Int64.Type}, {"MAT_BU_LAM", Int64.Type}, {"MAT_BU_DECO", Int64.Type}, {"MAT_BU_IND", Int64.Type}, {"MAT_BU_CAST", Int64.Type}, {"MAT_DIM_PLF", Int64.Type}, {"MAT_DIM_DLF", Int64.Type}, {"MAT_DIM_MF", Int64.Type}, {"MAT_PACK_BOXE", Int64.Type}, {"MAT_PACK_LOOSE", Int64.Type}, {"MAT_PACK_ENDCAP", Int64.Type}, {"MAT_LT_STX", Int64.Type}, {"MAT_LT_ST1", Int64.Type}, {"MAT_LT_NLT", Int64.Type}, {"LEAD_TIME", Int64.Type}, {"MIX_LT", Int64.Type}, {"MAT_PLATFORM", Int64.Type}}),
    #"Filtered Rows2" = Table.SelectRows(#"Changed Type", each ([SHIPMENT_CONDITION_DESCR] <> "AGC Sea")),
    #"Replaced Value1" = Table.ReplaceValue(#"Filtered Rows2",",",".",Replacer.ReplaceText,{"CUST_NAME1"}),
    #"Filtered Rows1" = Table.SelectRows(#"Replaced Value1", each [DELIVERY_NB] <> "8001201442"),

```
    #"Renamed Columns1" = Table.RenameColumns(#"Filtered Rows1",{{"MAT_BU_TOTAL",
"TOTAL_BU"}})
in
    #"Renamed Columns1"
```

Annexe 4 – Descriptif des variables

Nom de la variable	Descriptif de la variable	Type de variable	Valeur de la variable
COMBINED_BU	S'il y a plusieurs business unit présente dans une delivery ou non	Binaire	0 ou 1
COMBINED_DIM	S'il y a plusieurs dimensions de verre présente dans une delivery ou non	Binaire	0 ou 1
COMBINED_FLOW	S'il y a plusieurs moyens de transport dans une delivery ou non	Binaire	0 ou 1
COMBINED_LT	S'il y a un lead time de fabrication différent pour le verre présent dans une delivery ou non	Binaire	0 ou 1
COMBINED_PACK	S'il y a plusieurs emballages présents dans une delivery ou non	Binaire	0 ou 1
COMPLEXITY	Le pourcentage de commande avec un lead time de fabrication de 5 jours par client	Quantitative continue	[0 ; 100]
CUST_CLUSTER_DESCR	La région dans laquelle le client est situé	Quantitative discrète	0, 1, 2, ..., 8
CUST_COUNTRY_NAME	Le pays dans lequel le client est situé	Quantitative discrète	0, 1, 2, ..., 44
CUST_GROUP_DESCR	L'entreprise mère à laquelle appartient le client		
CUST_NAME1	Le nom du client (Indexé de 0 à 1456)	Quantitative discrète	0, 1, 2, ..., 1456
CUST_TYPE_DESCR	Le type de client (Exporter ou client normal	Quantitative discrète	0, 1
DELIVERY_NB	Le numéro unique pour chaque delivery crée dans le système SAP	Quantitative discrète	8001087058 8001093881 ....
FACTORY_CLUSTER	Région dans laquelle se trouve l'usine	Quantitative discrète	
FACTORY_CODE	Le code qui représente l'usine	Quantitative continue	1,2, 3...,12
FLOW_CBL	Lorsque le camion d'expédition à 2 points de chargement et 1 destinataire ou non	Binaire	0 ou 1
FLOW_CBLU	Lorsque le camion d'expédition à 2 points de chargement et 2 destinataires ou non	Binaire	0 ou 1
FLOW_CBU	Lorsque le camion d'expédition à 1 points de chargement et 2 destinataires ou non	Binaire	0 ou 1

FLOW_FTL	Lorsque le camion d'expédition est complet avec une seule delivery ou non	Binaire	0 ou 1
FLOW_GROUPAGE	Lorsque le camion d'expédition contient plusieurs delivery ou non	Binaire	0 ou 1
FREQUENCE	La fréquence de commande d'un client par an	Quantitative continue	1 à 4201
FREQUENCE_MONTH	La fréquence de commande d'un client par mois	Quantitative continue	0,066 à 280,33
INSTABILITY	Le pourcentage de commande modifiée par client	Quantitative continue	[0 ; 100]
LEAD_TIME	Le temps entre la prise de commande et la livraison (Sans les weekends et jour férié)	Quantitative continue	0, 1, ...,135
MAT_BU_CAST	Lorsqu'il y a du verre cast/moulé dans la delivery ou non	Binaire	0 ou 1
MAT_BU_COATING	Lorsqu'il y a du verre coaté dans la delivery ou non	Binaire	0 ou 1
MAT_BU_DECO	Lorsqu'il y a du verre de décoration dans la delivery ou non	Binaire	0 ou 1
MAT_BU_FLOAT	Lorsqu'il y a du verre floaté dans la delivery ou non	Binaire	0 ou 1
MAT_BU_IND	Lorsqu'il y a du verre industriel dans la delivery ou non	Binaire	0 ou 1
MAT_BU_LAM	Lorsqu'il y a du verre laminé dans la delivery ou non	Binaire	0 ou 1
MAT_DIM_DLF	Lorsqu'il y a du verre de dimension 3m * 3.15m (Demi-largeur Fixe) ou non	Binaire	0 ou 1
MAT_DIM_MF	Lorsqu'il y a du verre de dimension personnalisée ou non	Binaire	0 ou 1
MAT_DIM_PLF	Lorsqu'il y a du verre de dimension 6m * 3.15m (Jumbo Size) ou non	Binaire	0 ou 1
MAT_LT_NLT	Lorsqu'il n'y a pas de lead time spécifique pour le verre ou non	Binaire	0 ou 1
MAT_LT_ST1	Lorsque le lead time pour fabriquer le verre est de 5 semaines ou non	Binaire	0 ou 1
MAT_LT_STX	Lorsque le lead time pour fabriquer le verre est de 5 jours ou non	Binaire	0 ou 1
MAT_MIXED_BLOCK_BL	Lorsqu'il y a plusieurs types de verre sur un même chevalet ou non	Binaire	0 ou 1

MAT_MIXED_END_CAP_BL	Lorsqu'il y a plusieurs types de verre dans un même emballage en bois ou non	Binaire	0 ou 1
MAT_MIXED_ORDER_BL	Lorsqu'il y a plusieurs types de verre dans un même camion ou non	Binaire	0 ou 1
MAT_NUMBER	Le nombre de type de verre différente sur une même delivery	Quantitative continue	0, 1, ..., 48
MAT_PACK_BOXE	Si le verre a été emballer ou non	Binaire	0 ou 1
MAT_PACK_ENDCAP	Si le verre a été emballé dans une caisse en bois ou non	Binaire	0 ou 1
MAT_PACK_LOOSE	Si le verre est en vrac sur les chevalets ou non	Binaire	0 ou 1
MAT_PLATFORM	Si une plateforme (petit chevalet) a été utilisée ou non	Binaire	0 ou 1
MAT_STILLAGE	Si un chevalet a été utilisé ou non	Binaire	0 ou 1
MIX_LT	Si la delivery comporte différent type de lead time comme STX, ST1 et NLT ou non	Binaire	0 ou 1
MODIF_DAY	Le jour de la semaine ou la delivery a été modifiée	Quantitative discrète	0,1,2,3,4
MODIF_DUE_AGC	Si la modification de la delivery est due à AGC ou non	Binaire	0 ou 1
MODIF_DUE_CUST	Si la modification de la delivery est due au client ou non	Binaire	0 ou 1
MODIF_MONTH	Le jour de l'année ou la delivery a été modifiée	Quantitative discrète	0, 1, ..., 11
MODIF_NUMBER	Le nombre de fois que la delivery a été modifiée	Quantitative discrète	0, 1, ..., 38
MODIF_NUMBER_T	Le nombre de fois que la delivery a été modifiée sans compter la modification faite après J-2	Quantitative discrète	0, 1, ..., 37
MODIF_OR_NO	Si la delivery a été modifiée ou non	Binaire	0 ou 1
MODIF_OR_NO_T	Si la delivery a été modifiée sans compter la modification à J-2	Binaire	0 ou 1
MODIF_QUARTER	Le quadrimestre ou la delivery a été modifiée	Quantitative discrète	0, 1, 2
MODIF_WEEK	La semaine ou la delivery a été modifiée	Quantitative discrète	0, 1, 2, ..., 41
MODIF_WITHIN_2DAY	Si la commande a été modifiée endéans les 2 jours après la création de la delivery	Binaire / Variable dépendante	0 ou 1
SHIP_LAST_DTTI	La date ou la delivery a été expédiée	Ordinale	01/01/2023 28/03/2024

SHIP_ORDER_DTTI	La date ou le client a pris la commande	Ordinale	01/01/2023 28/03/2024
SHIP_PROJECT_CODE	Si la delivery fait partie d'un contrat avec un certain client ou non	Binaire	0 ou 1
SHIPMENT_CONDITION_DESCR	Le moyen d'expédition de la delivery (Camion, bateau, avion, ...)	Quantitative discrète	1, 2, 3, ..., 6
SHIPPING_POINT_CODE	L'usine d'où la delivery est expédiée	Quantitative discrète	1, 2, 3, 4, ..., 24
TOTAL_BU	Le nombre de business unit présente dans une delivery	Quantitative discrète	1, 2, 3, 4, 5, 6
TOTAL_DIM	Le nombre de verre de dimension différente présente dans une delivery	Quantitative discrète	1, 2, 3
TOTAL_FLOW	Le nombre de verre de dimension différente présent dans une delivery	Quantitative discrète	1, 2, 3
TOTAL_LT	Le nombre de verre avec un lead time de fabrication différent présent dans une delivery	Quantitative discrète	1, 2, 3
TOTAL_PACK	Le nombre de verre avec un emballage différent présent dans une delivery	Quantitative discrète	1, 2, 3

Annexe 5 – Code Python

Annexe 5.1 - Transformation variables catégorielles > 100 catégories :

```
import pandas as pd

# Charger le dataset en utilisant le séparateur correct (si nécessaire)
df = pd.read_csv('Table2.csv', sep=';')

# Créer une fonction pour mapper chaque catégorie unique à un entier
def encode_column(column):
    unique_values = column.unique()
    value_to_int = {value: idx for idx, value in enumerate(unique_values)}
    return column.map(value_to_int)

# Appliquer la fonction d'encodage pour chaque colonne catégorielle
df['CUST_NAME1'] = encode_column(df['CUST_NAME1'])
df['CUST_COUNTRY_NAME'] = encode_column(df['CUST_COUNTRY_NAME'])
df['CUST_GROUP_DESCR'] = encode_column(df['CUST_GROUP_DESCR'])


# Sauvegarder le DataFrame avec les modifications dans le fichier CSV original
df.to_csv('Table2.csv', sep=';', index=False)

# Afficher la matrice de corrélation
print("Transformation des variables CUST_NAME1, CUST_GROUP_DESCR")
```

Annexe 5.2 - Transformation variables catégorielles < 100 catégories :

```
import pandas as pd

# Chemin vers le fichier CSV
file_path = 'Table2.csv'

# Lire le fichier CSV dans un DataFrame
df = pd.read_csv(file_path, delimiter=';')  # Ajustez le délimiteur si nécessaire

# Remplacer la virgule par un point pour les colonnes 'COMPLEXITY' et 'INSTABILITY'
df['COMPLEXITY'] = df['COMPLEXITY'].astype(str).str.replace(',', '.').astype(float)
df['INSTABILITY'] = df['INSTABILITY'].astype(str).str.replace(',', '.').astype(float)
df['FREQUENCE_MONTH'] = df['FREQUENCE_MONTH'].astype(str).str.replace(',',
'.').astype(float).round(1)

# Sauvegarder le DataFrame avec les modifications dans le fichier CSV original
df.to_csv('Table2.csv', sep=';', index=False)

# Afficher la matrice de corrélation
```

```
print("Transformation des variables string")
```

Annexe 5.3 - Transformation des séparateurs de nombre à virgule :

```
import pandas as pd
from sklearn.cluster import KMeans

# Chargement de vos données
# Assurez-vous de remplacer 'chemin_vers_votre_fichier.csv' par le chemin réel de votre fichier CSV
data = pd.read_csv('Table2.csv', sep=';')

# Sélection des variables pour le clustering
X = data[["COMPLEXITY", "INSTABILITY","FREQUENCE_MONTH"]]

# Création et ajustement du modèle KMeans avec 3 clusters
model = KMeans(n_clusters=3, random_state=42)
data['CLUSTER2'] = model.fit_predict(X)

# Sauvegarde des résultats dans un nouveau fichier CSV sans écraser le fichier original
# Spécifiez un nouveau nom de fichier pour enregistrer le résultat
data.to_csv('Table2.csv', index=False, sep=';')

print("Clustering réalisé et nouveau fichier CSV créé avec les clusters.")
```

Annexe 5.4 - Script Python pour les corrélations entre les variables

```
import pandas as pd

import seaborn as sns
import matplotlib.pyplot as plt


# Load your dataset
df = pd.read_csv('table.csv')


# Calculate the correlation matrix
correlation_matrix = df.corr()


# Extract the correlation coefficients with the target variable
target_variable = 'MODIF_WITHIN_2DAY'
correlations = correlation_matrix[target_variable].drop(target_variable)


# Sort the correlations
```

```
correlations_sorted = correlations.sort_values(ascending=False)


# Plotting
plt.figure(figsize=(10, 8))
sns.barplot(x=correlations_sorted.values, y=correlations_sorted.index, palette='coolwarm',
edgecolor='black')
plt.title(f'Corrélations de la Variable "{target_variable}" avec les Variables Indépendantes')
plt.xlabel('Coefficient de Corrélation')
plt.ylabel('Variable')
plt.show()
```

Annexe 5.5 - Modèle de régression logistique

```
#Logistics Regression - Seuil Personnalisé
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Définissez le seuil ici
seuil_personnalise = 0.35# Modifiez cette valeur selon votre besoin

# Charger les données
data = pd.read_csv('Table2.csv', delimiter=';')

# Sélectionner les variables explicatives et la variable cible
X = data[["MODIF_OR_NO_T", "LEAD_TIME", "MODIF_NUMBER_T",
"FREQUENCE_MONTH", "INSTABILITY",
"MAT_NUMBER","COMBINED_BU","COMPLEXITY"]]
y = data['MODIF_WITHIN_2DAY']

# Séparer les données en ensembles d'entraînement et de test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardiser les features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Créer et entraîner le modèle de régression logistique
model = LogisticRegression()
model.fit(X_train_scaled, y_train)
```

```
# Obtenir les probabilités de la classe positive
y_probs = model.predict_proba(X_test_scaled)[:, 1]

# Appliquer le seuil personnalisé pour obtenir les nouvelles prédictions
y_pred_custom_threshold = [1 if prob > seuil_personnalise else 0 for prob in y_probs]

# Évaluer le modèle avec le seuil personnalisé
print(f"Accuracy avec seuil de {seuil_personnalise}:", accuracy_score(y_test,
y_pred_custom_threshold))
print(f"Confusion Matrix avec seuil de {seuil_personnalise}:\n", confusion_matrix(y_test,
y_pred_custom_threshold))
print(f"Classification Report avec seuil de {seuil_personnalise}:\n", classification_report(y_test,
y_pred_custom_threshold))
```

Annexe 5.6 - Modèle de régression logistique + SMOTE

```
# Logistic Regression - SMOTE with Custom Threshold
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from imblearn.over_sampling import SMOTE

# Définissez le seuil ici
seuil_personnalise = 0.35  # Modifiez cette valeur selon votre besoin

# Charger les données
data = pd.read_csv('Table2.csv', delimiter=';')

# Sélectionner les variables explicatives et la variable cible
X = data[["MODIF_OR_NO_T", "LEAD_TIME", "MODIF_NUMBER_T",
"FREQUENCE_MONTH", "INSTABILITY", "MAT_NUMBER", "COMBINED_BU",
"COMPLEXITY"]]
y = data['MODIF_WITHIN_2DAY']

# Séparer les données en ensembles d'entraînement et de test
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Appliquer SMOTE pour équilibrer les classes dans l'ensemble d'entraînement
smote = SMOTE(random_state=42)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)

# Standardiser les features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train_resampled)
X_test_scaled = scaler.transform(X_test)

# Créer et entraîner le modèle de régression logistique
model = LogisticRegression()
model.fit(X_train_scaled, y_train_resampled)

# Obtenir les probabilités de la classe positive
y_probs = model.predict_proba(X_test_scaled)[:, 1]

# Appliquer le seuil personnalisé pour obtenir les nouvelles prédictions
y_pred_custom_threshold = [1 if prob > seuil_personnalise else 0 for prob in y_probs]

# Évaluer le modèle avec le seuil personnalisé
print(f"Accuracy avec seuil de {seuil_personnalise}:", accuracy_score(y_test,
y_pred_custom_threshold))
print(f"Confusion Matrix avec seuil de {seuil_personnalise}:\n", confusion_matrix(y_test,
y_pred_custom_threshold))
print(f"Classification Report avec seuil de {seuil_personnalise}:\n", classification_report(y_test,
y_pred_custom_threshold))
```

Annexe 5.7 - Modèle de régression logistique + Seuil optimal

```
#Logistics Regression - Seuil Automatique
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_curve, f1_score, precision_recall_curve
```

```
# Charger les données
data = pd.read_csv('Table2.csv', delimiter=';')

# Sélectionner les variables explicatives et la variable cible
X = data[["MODIF_OR_NO_T", "LEAD_TIME", "MODIF_NUMBER_T",
"FREQUENCE_MONTH", "INSTABILITY",
"MAT_NUMBER","COMBINED_BU","COMPLEXITY"]]
y = data['MODIF_WITHIN_2DAY']

# Séparer les données en ensembles d'entraînement et de test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardiser les features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Créer et entraîner le modèle de régression logistique
model = LogisticRegression()
model.fit(X_train_scaled, y_train)

# Obtenir les probabilités de la classe positive
y_probs = model.predict_proba(X_test_scaled)[:, 1]

# Calculer les courbes de précision-rappel et trouver le meilleur seuil pour le score F1
precision, recall, thresholds = precision_recall_curve(y_test, y_probs)
fscore = (2 * precision * recall) / (precision + recall)
ix = np.argmax(fscore)
print('Meilleur Seuil=%f, Score F1=%f' % (thresholds[ix], fscore[ix]))

# Appliquer le seuil optimal pour obtenir les nouvelles prédictions
y_pred_optimal = [1 if prob > thresholds[ix] else 0 for prob in y_probs]

# Calculer et afficher la matrice de confusion
from sklearn.metrics import confusion_matrix
conf_matrix = confusion_matrix(y_test, y_pred_optimal)
print("Matrice de confusion avec seuil optimal:\n", conf_matrix)
```

Annexe 5.8 - Modèle de régression logistique + sous échantillonnage

```
# Logistics Regression - Seuil Personnalisé avec Sous-échantillonnage

import pandas as pd

import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

```
from imblearn.under_sampling import RandomUnderSampler

# Définissez le seuil ici
seuil_personnalise = 0.35  # Modifiez cette valeur selon votre besoin

# Charger les données
data = pd.read_csv('Table2.csv', delimiter=';')

# Sélectionner les variables explicatives et la variable cible
X = data[["MODIF_OR_NO_T", "LEAD_TIME", "MODIF_NUMBER_T",
"FREQUENCE_MONTH", "INSTABILITY", "MAT_NUMBER", "COMBINED_BU",
"COMPLEXITY"]]
y = data['MODIF_WITHIN_2DAY']

# Séparer les données en ensembles d'entraînement et de test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Appliquer le sous-échantillonnage pour équilibrer les classes dans l'ensemble d'entraînement
undersampler = RandomUnderSampler(random_state=42)
X_train_resampled, y_train_resampled = undersampler.fit_resample(X_train, y_train)

# Standardiser les features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train_resampled)
X_test_scaled = scaler.transform(X_test)

# Créer et entraîner le modèle de régression logistique
model = LogisticRegression()
model.fit(X_train_scaled, y_train_resampled)

# Obtenir les probabilités de la classe positive
y_probs = model.predict_proba(X_test_scaled)[:, 1]

# Appliquer le seuil personnalisé pour obtenir les nouvelles prédictions
y_pred_custom_threshold = [1 if prob > seuil_personnalise else 0 for prob in y_probs]

# Évaluer le modèle avec le seuil personnalisé
```

```
print(f"Accuracy avec seuil de {seuil_personnalise}:", accuracy_score(y_test,
y_pred_custom_threshold))
print(f"Confusion Matrix avec seuil de {seuil_personnalise}:\n", confusion_matrix(y_test,
y_pred_custom_threshold))
print(f"Classification Report avec seuil de {seuil_personnalise}:\n", classification_report(y_test,
y_pred_custom_threshold))
```

Annexe 5.9 - Modèle du réseau de neurones

```
#Neural Network
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report, confusion_matrix
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.utils import to_categorical

# Charger les données
data = pd.read_csv('Table2.csv', sep=';')

# Correction de la liste des variables
variables = ["MODIF_OR_NO_T", "LEAD_TIME", "MODIF_NUMBER_T",
"FREQUENCE_MONTH", "INSTABILITY",
"MAT_NUMBER","COMBINED_BU","COMPLEXITY"]
data_selectionnee = data[variables + ['MODIF_WITHIN_2DAY']].fillna(data.median())

# Convertir les variables catégorielles en variables numériques

data_selectionnee['MODIF_DAY'] = data_selectionnee['MODIF_DAY'].astype('category').cat.codes
data_selectionnee['CLUSTER'] = data_selectionnee['CLUSTER'].astype('category').cat.codes


# Préparer les données d'entraînement et de test
X = data_selectionnee.drop('MODIF_WITHIN_2DAY', axis=1)
y = data_selectionnee['MODIF_WITHIN_2DAY']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Normalisation des caractéristiques
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Convertir les étiquettes en format catégoriel
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
```

```
# Construire le modèle
model = Sequential([
    Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
    Dense(64, activation='relu'),
        Dense(y_train.shape[1], activation='softmax')  # Utilisation dynamique du nombre de classes
])

# Compiler le modèle
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Entraîner le modèle
model.fit(X_train, y_train, epochs=100, batch_size=32, verbose=1)

# Prédire les classes sur l'ensemble de test
y_pred_prob = model.predict(X_test)
y_pred = np.argmax(y_pred_prob, axis=1)

# Récupérer les vraies étiquettes de y_test
y_true = np.argmax(y_test, axis=1)

# Afficher le rapport de classification et la matrice de confusion
print(classification_report(y_true, y_pred))
print("Matrice de confusion :")
print(confusion_matrix(y_true, y_pred))

# Évaluer le modèle
loss, accuracy = model.evaluate(X_test, y_test, verbose=0)
print(f'Précision du test: {accuracy:.3f}')
```

Annexe 5.10 - Modèle de la forêt aléatoire

```
#Random Forest
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score

# Chargement des données
data = pd.read_csv('Table2.csv', sep=';')

# Sélection des variables explicatives et de la variable cible
```

```
features = ["MODIF_OR_NO_T", "LEAD_TIME", "MODIF_NUMBER_T",
"FREQUENCE_MONTH", "INSTABILITY",
"MAT_NUMBER","COMBINED_BU","COMPLEXITY"]
X = data[features]
y = data['MODIF_WITHIN_2DAY']

# Division des données en ensembles d'entraînement et de test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)

# Normalisation des données
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Entraînement du modèle Random Forest
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train_scaled, y_train)

# Prédiction sur l'ensemble de test
y_pred_rf = rf_model.predict(X_test_scaled)
y_pred_proba_rf = rf_model.predict_proba(X_test_scaled)[:, 1]

# Évaluation du modèle
print("Classification report:")
print(classification_report(y_test, y_pred_rf))
print("AUC:", roc_auc_score(y_test, y_pred_proba_rf))

# Affichage de la matrice de confusion
conf_matrix_rf = confusion_matrix(y_test, y_pred_rf)
sns.heatmap(conf_matrix_rf, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Random Forest Confusion Matrix')
plt.show()
```

Annexe 5.11 - Modèle de la forêt aléatoire + hyperparamètre

```
#Random Forest 2 - Hyperparameter
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score

# Chargement des données
```

```
data = pd.read_csv('Table2.csv', sep=';')

# Sélection des variables explicatives et de la variable cible
features = ["MODIF_OR_NO_T", "LEAD_TIME", "MODIF_NUMBER_T",
"FREQUENCE_MONTH", "INSTABILITY",
"MAT_NUMBER","COMBINED_BU","COMPLEXITY"]
X = data[features]
y = data['MODIF_WITHIN_2DAY']

# Division des données en ensembles d'entraînement et de test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)

# Normalisation des données
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Paramètres pour GridSearchCV
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'bootstrap': [True, False]
}

# Création du modèle Random Forest avec GridSearchCV
rf = RandomForestClassifier(random_state=42)
grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, cv=3, n_jobs=-1, verbose=2,
scoring='roc_auc')
grid_search.fit(X_train_scaled, y_train)

# Meilleurs paramètres et modèle
best_params = grid_search.best_params_
best_rf_model = grid_search.best_estimator_

# Prédiction sur l'ensemble de test avec le meilleur modèle
y_pred_rf = best_rf_model.predict(X_test_scaled)
y_pred_proba_rf = best_rf_model.predict_proba(X_test_scaled)[:, 1]

# Évaluation du modèle avec les meilleurs paramètres
print("Meilleurs hyperparamètres:", best_params)
print("Classification report:")
print(classification_report(y_test, y_pred_rf))
print("AUC:", roc_auc_score(y_test, y_pred_proba_rf))

# Affichage de la matrice de confusion avec les meilleurs paramètres
conf_matrix_rf = confusion_matrix(y_test, y_pred_rf)
```

```
sns.heatmap(conf_matrix_rf, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Random Forest Confusion Matrix with Best Parameters')
plt.show()
```

Annexe 6 – Résultats de tous les modèles

	Regression Logistique	Accuracy	True Positive	True Negative	False Positive	False Negative
1	Seuil = 0,35	0,92	257	13948	236	972
2	Seuil = 0,2	0,9	438	13469	715	791
3	Seuil = 0,5	0,92	156	14089	95	1073
4	Seuil Optimal	0,92	547	13132	1052	682
5	SMOTE - Seuil 0,35	0,64	1038	8961	5223	191
6	SMOTE - Seuil 0,5	0,77	837	11153	3031	392
7	SMOTE - Seuil 0,2	0,77	837	11153	3031	392
8	SMOTE - Seuil 0,7	0,88	531	13041	1143	698
9	Undersample - Seuil 0,5	0,76	881	10968	3216	348
10	Undersample - Seuil 0,35	0,62	1064	8485	5699	165
11	Undersample - Seuil 0,7	0,87	570	12962	1222	659

Annexe 7 – Calcul temps de préparation d'une commande

Annexe 7.1 – Temps par action :

Etape de la préparation de commande	Catégorie	% des commandes	Moyenne du temps/Min
1) Localisation du verre	Pas de différence	100%	5
2) Collecte du verre	BU = 1	73,66%	15
	BU = 2	19,53%	25
	BU > 3	6,81%	40
4) Rassemblement au point de préparati	Pas de différence	100%	10
3) Découpe	PLF	67,08%	20
	DLF	32,92%	0
4) Emballage	Vrac	89,02%	10
	Boxe ou Endcaps	10,98%	15
5) Etiquetage et documentation	Pas de différence	100%	10
6) Transport au quai de livraison	Pas de différence	100%	10

Annexe 7.2 - Temps par type de commande :

Descriptif de la commande	Temps/ Min
BU1 + PLF + Vrac	80
BU1 + PLF + Boxe ou Endcaps	85
BU1 + DLF+ Vrac	60
BU1 + DLF+ Boxe ou Endcaps	65
BU2 + PLF + Vrac	90
BU2 + PLF + Boxe ou Endcaps	95
BU2 + DLF+ Vrac	70
BU2 + DLF+ Boxe ou Endcaps	75
BU3 + PLF + Vrac	105
BU3 + PLF + Boxe ou Endcaps	110
BU3 + DLF+ Vrac	85
BU3 + DLF+ Boxe ou Endcaps	90