

巨量資料分析 Homework2

巴斯克乳酪蛋糕

統計 111

H24076029

劉米婷

統計 111

H24071215

陳柔漪

統計 111

H24071273

林少穎

Content

1. Introduction.....	2
1.1 缺失值的比例.....	2
1.2 各個特徵對 Y 的關係.....	5
1.3 資料分布.....	7
2. Methodology	10
2.1 刪除資訊量少的資料.....	10
2.2 轉換類別資料，進行 label encoding	10
2.3 使用相關性填補遺失值.....	11
2.4 填補遺失值.....	11
2.5 模型預測.....	12
3. Experimental analysis.....	13
3.1 方法一.....	13
3.2 方法二.....	16
3.3 方法三.....	19
3.4 方法四.....	22
3.5 方法五.....	25
3.6 方法比較.....	27
4. Conclusion.....	28
5. Citation.....	28

1. Introduction

1.1 缺失值的比例

本次作業中有七個資料集，分別有各自的訓練資料與測試資料，而訓練資料的特徵數量與資料筆數如下表：

	Data1	Data2	Data3	Data4	Data5	Data6	Data7
特徵數量	8	8	13	8	4	11	6
資料筆數	765	574	371	6064	7077	1187	231

每個資料集的每個特徵都有不同數量的遺失值，我們將每筆訓練資料各個特徵的缺失值數量與缺失值於此特徵的比例，如下表所述，可以發現所有特徵遺失值的比例大約都落在 27%至 33%之間：

特徵	1	2	3	4	5	6	7	8
遺失數量	234	220	218	225	219	239	249	237
遺失比例	30.6%	28.8%	28.5%	29.4%	28.6%	31.2%	32.6%	31.0%

▲ Data1

特徵	1	2	3	4	5	6	7	8
遺失數量	167	161	195	179	151	163	167	173
遺失比例	29.1%	28.1%	34.0%	31.1%	26.3%	28.4%	29.1%	30.1%

▲ Data2

特徵	1	2	3	4	5	6	7	8
遺失數量	110	124	103	103	105	118	124	122
遺失比例	29.7%	33.4%	27.8%	27.8%	28.3%	31.8%	33.4%	32.9%
特徵	9	10	11	12	13			
遺失數量	118	105	104	110	102			
遺失比例	31.8%	28.3%	28.0%	30.0%	27.5%			

▲ Data3

特徵	1	2	3	4	5	6	7	8
遺失數量	1755	1864	1834	1811	1764	1789	1850	1893
遺失比例	28.9%	30.7%	30.2%	30.0%	29.1%	29.5%	30.5%	31.22%

▲ Data4

特徵	1	2	3	4
遺失數量	2132	2149	2089	2114
遺失比例	30.1%	30.4%	29.5%	30.0%

▲ Data5

特徵	1	2	3	4	5	6	7	8
遺失數量	367	395	362	357	351	327	342	367
遺失比例	30.9%	32.3%	30.5%	30.1%	29.6%	27.6%	28.8%	30.9%
特徵	9	10	11					
遺失數量	346	366	321					
遺失比例	29.2%	30.8%	27.0%					

▲ Data6

特徵	1	2	3	4	5	6
遺失數量	68	62	74	63	78	68
遺失比例	29.4%	26.8%	32.0%	27.3%	33.8%	29.4%

▲ Data7

以各筆資料來看，每個資料集都有缺少過多特徵的資料，如下表所述：

缺少 數量	0	1	2	3	4	5	6	7	8
資料 筆數	48	152	216	189	120	31	8	1	0
佔比	6.3%	20.0%	28.2%	24.7%	15.7%	4.1%	1.1%	0.1%	0.0%

▲ Data1

缺少 數量	0	1	2	3	4	5	6	7	8
資料 筆數	38	106	187	139	69	27	8	0	0
佔比	6.6%	18.5%	32.6%	24.2%	12.0%	4.7%	1.4%	0.0%	0.0%

▲ Data2

缺少 數量	0	1	2	3	4	5	6	7
資料 筆數	6	21	35	87	96	68	41	10
佔比	1.6%	5.7%	9.4%	23.5%	25.8%	18.3%	11.1%	2.7%
缺少 數量	8	9	10	11	12	13		
資料 筆數	7	0	0	0	0	0		
佔比	1.9%	0.0%	0.0%	0.0%	0.0%	0.0%		

▲ Data3

缺少 數量	0	1	2	3	4	5	6	7	8
資料 筆數	359	1180	1791	1565	822	277	61	8	1
佔比	5.9%	19.5%	29.5%	25.8%	13.6%	4.6%	1.0%	0.1%	0.02%

▲ Data4

缺少 數量	0	1	2	3	4
資料 筆數	1693	2960	1841	530	63
佔比	23.8%	41.8%	26.0%	7.5%	0.9%

▲ Data5

缺少 數量	0	1	2	3	4	5	6	7
資料 筆數	27	100	242	320	245	171	63	13
佔比	2.3%	8.4%	20.4%	27.0%	20.6%	14.4%	5.3%	1.1%
缺少 數量	8	9	10	11				
資料 筆數	3	2	0	1				
佔比	0.3%	0.2%	0.0%	0.1%				

▲ Data6

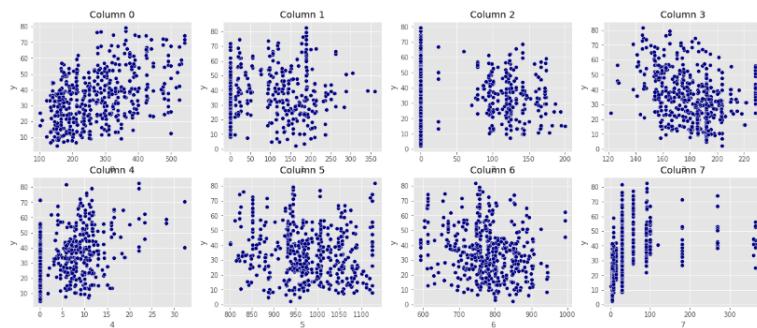
缺少 數量	0	1	2	3	4	5	6
資料 筆數	30	72	68	41	18	2	0
佔比	13.0%	31.2%	29.4%	17.8%	7.8%	0.9%	0.0%

▲ Data7

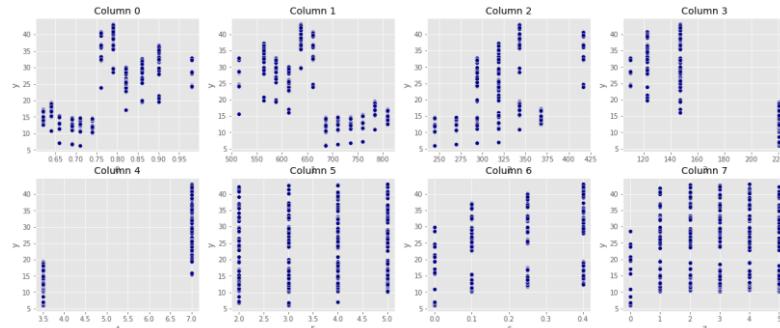
1.2 各個特徵對 Y 的關係

將每個訓練集的特徵值對 Y 的散佈圖畫出，觀察之間是否具有相關性。

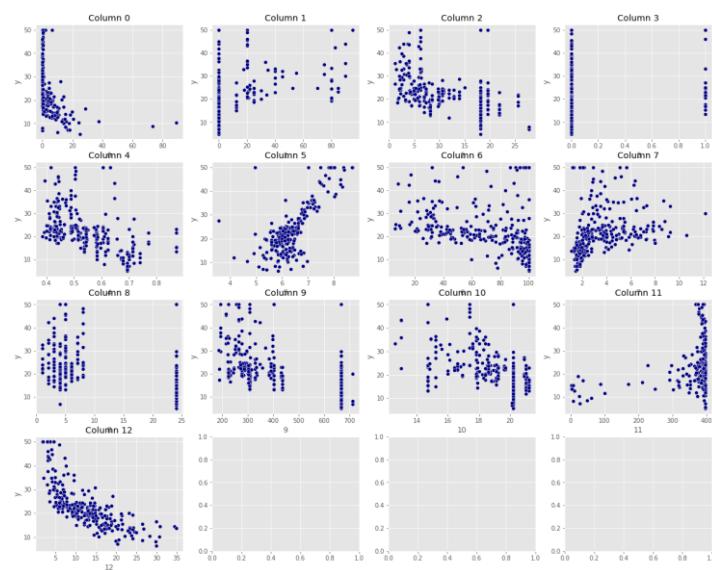
◆ Data1：Y 對 Column 0、4 有正相關。



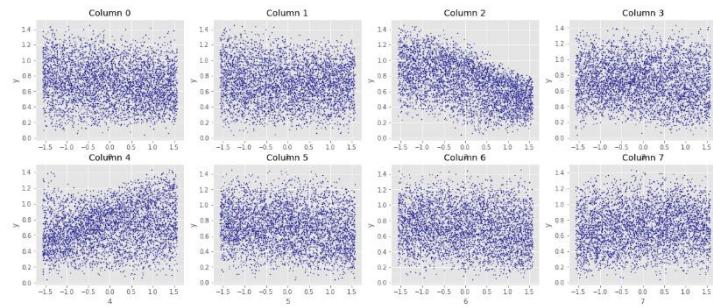
◆ Data2：Y 對 Column 0、4 有正相關。



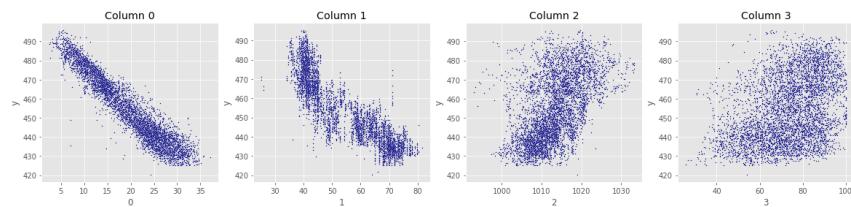
◆ Data3：Y 對 Column 5 有正相關，對 Column 12 有負相關。



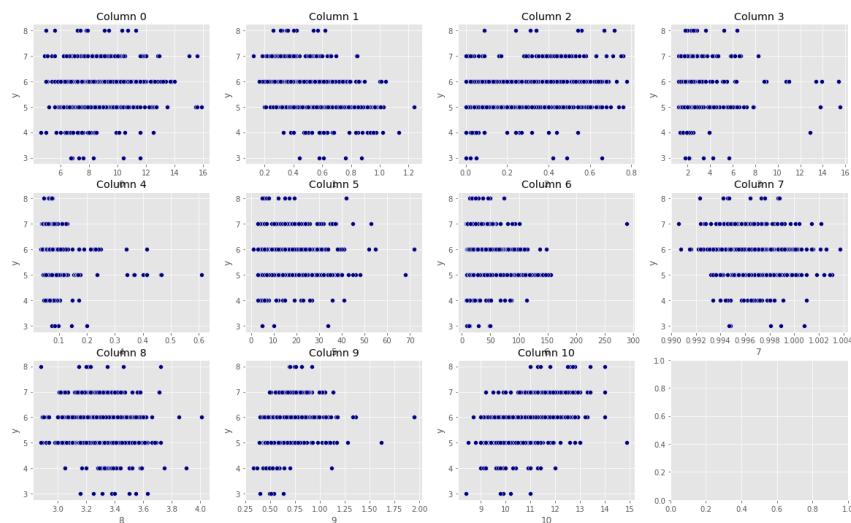
◆ Data4：Y 對 Column 4 有正相關，對 Column 2 有負相關。



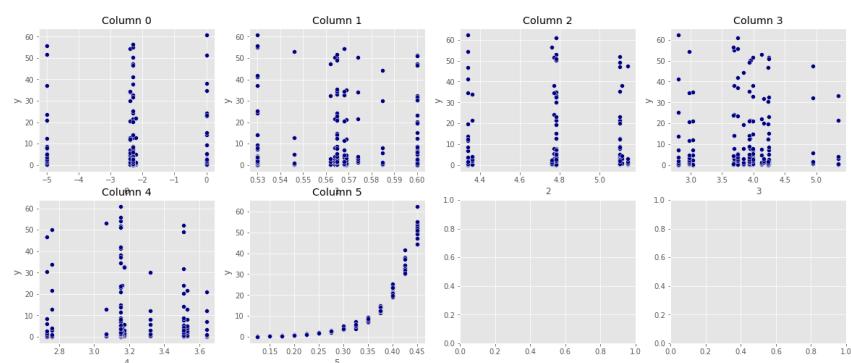
◆ Data5：Y 對 Column 2 有正相關，對 Column 0、1 有負相關。



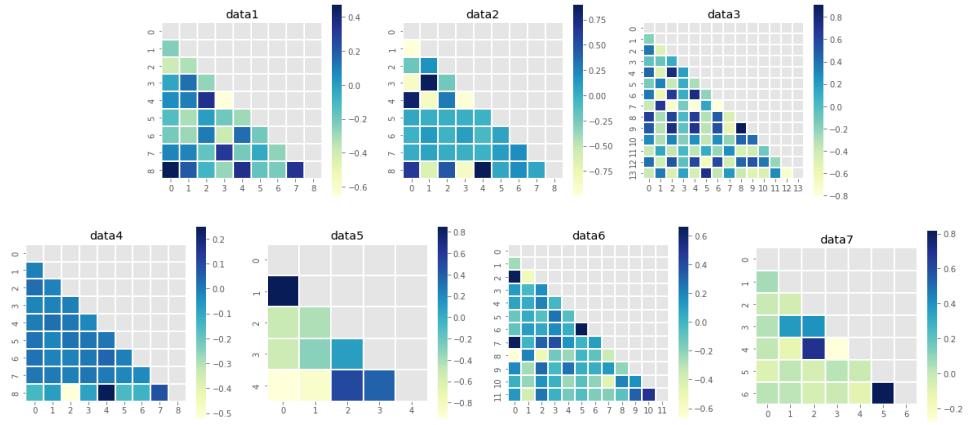
◆ Data6：Y 對 Column 10 有正相關。



◆ Data7：Y 對 Column 5 有正相關。

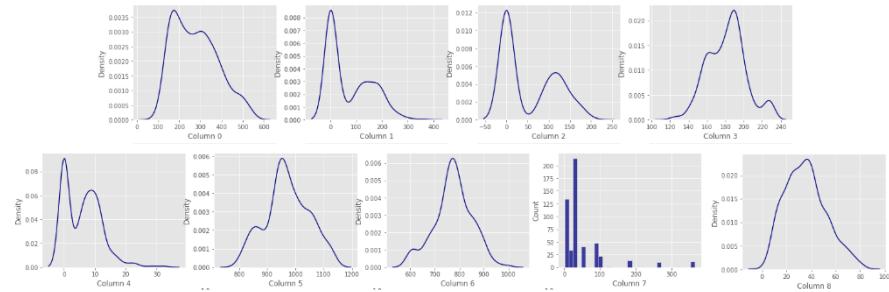


若將訓練集的各個特徵與 Y 的相關係數以熱圖呈現，則如下圖所示：

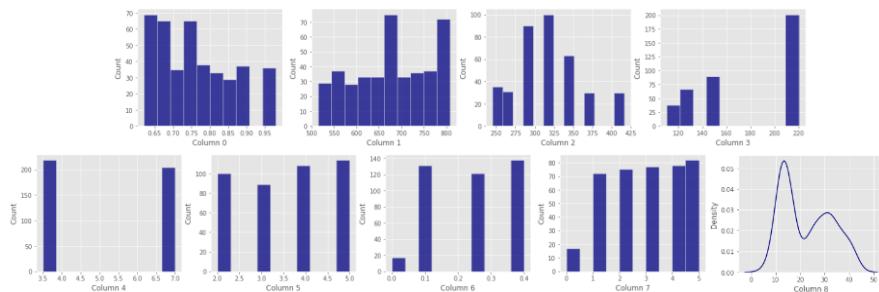


1.3 資料分布

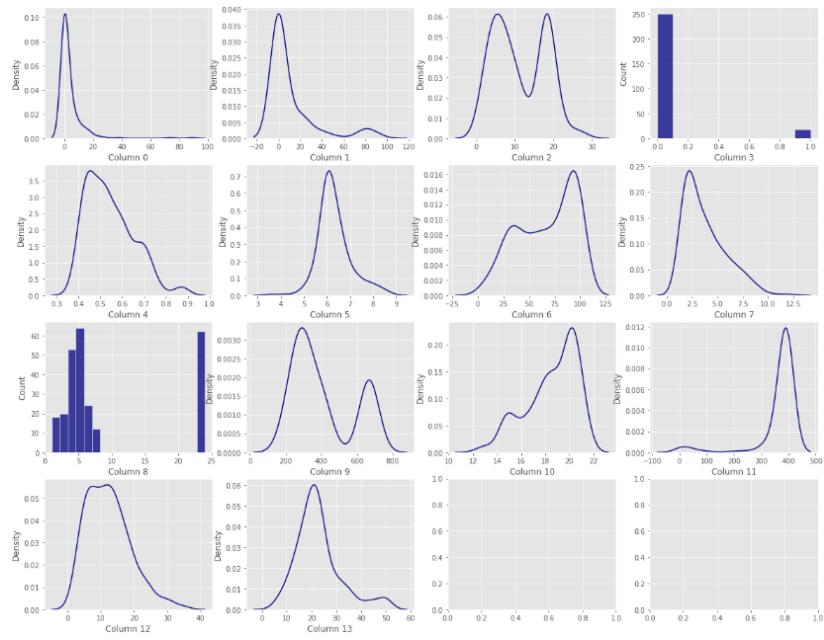
我們將每個訓練資料的特徵與 Y 的分布畫出，若資料的類別少於 20 種，則視為類別型以直方圖呈現，大於 20 種則視為連續型以機率密度圖呈現。



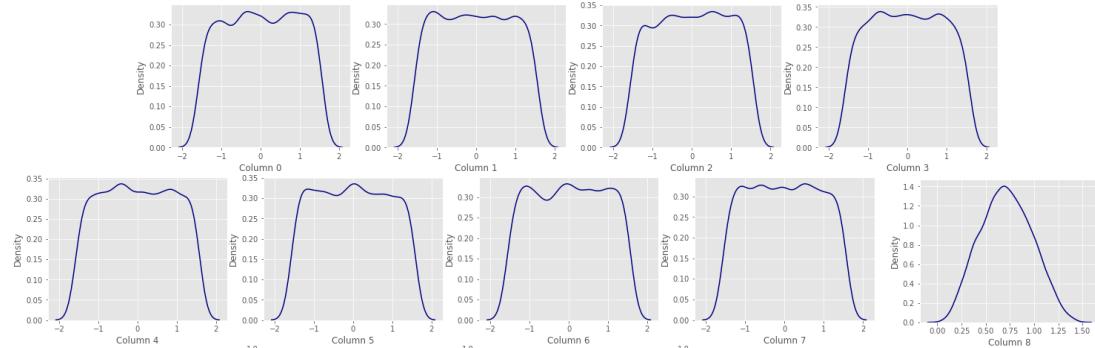
▲ Data1



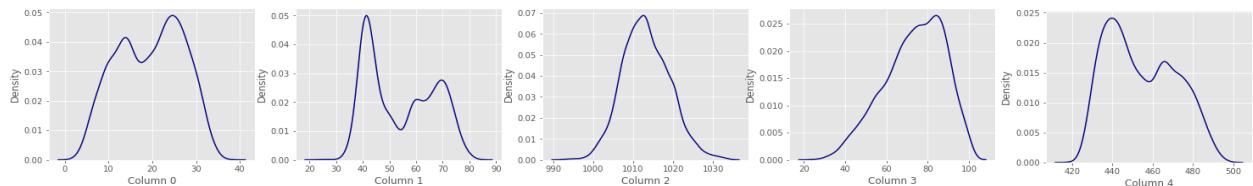
▲ Data2



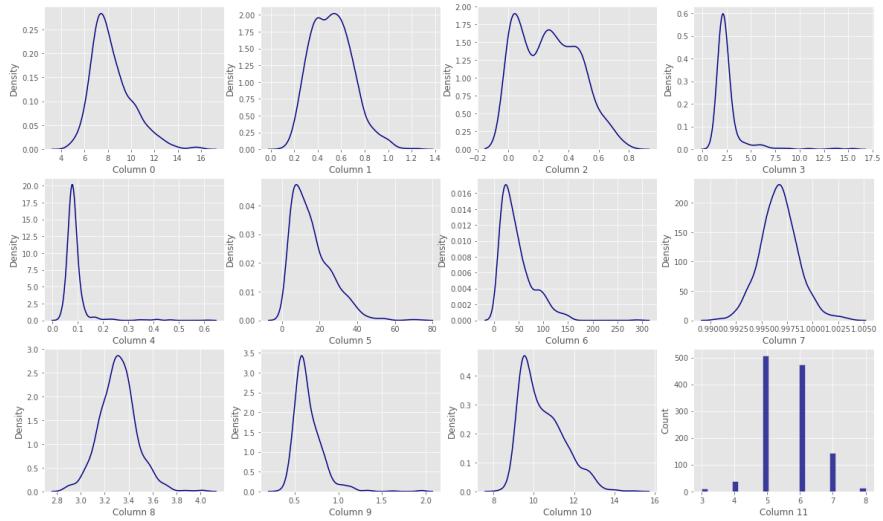
▲ Data3



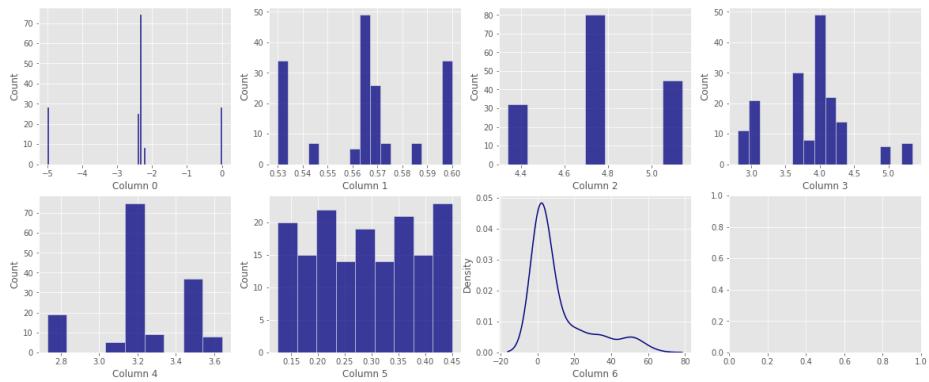
▲ Data4



▲ Data5



▲ Data6



▲ Data7

可以發現各個資料被判斷為類別型或連續型資料的數量都不盡相同，甚至有些全部皆為連續型，或有些全部皆為類別型。

2. Methodology

2.1 刪除資訊量少的資料

將每個訓練資料集，若一筆資料缺少的特徵占比超過 10%，則將此刪除，僅留下資訊量較多的訓練資料，刪除前後的資料筆數如下表所述：

	Data1	Data2	Data3	Data4	Data5	Data6	Data7
刪除前	765	574	371	6064	7077	1187	231
刪除後	725	539	354	5717	6484	1105	211

2.2 轉換類別資料，進行 label encoding

若一特徵值的類別種類少於一特定值（閥值），我們將此特徵視為類別的順序型變數，並做 label encoding，而閥值嘗試設為 10 與 20，各訓練集需轉換為類別型的特徵如下表所示：

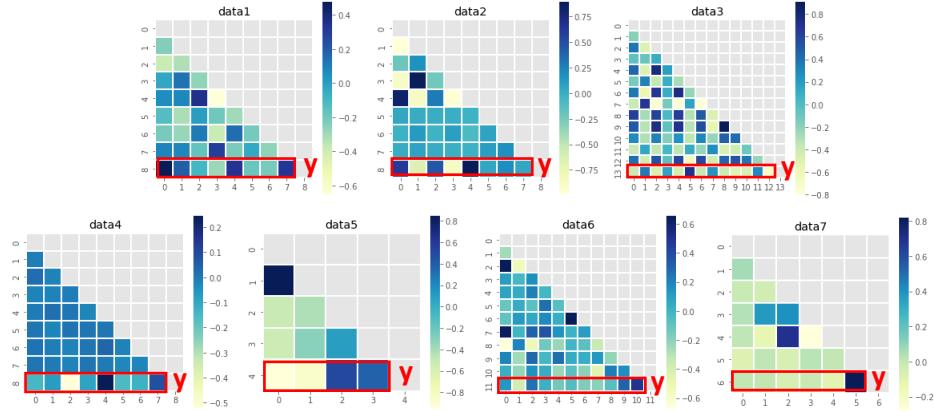
閥值=10	類別型特徵(Column)	連續型特徵(Column)
Data1		0,1,2,3,4,5,6,7,8
Data2	2,3,4,5,6,7	0,1,8
Data3	3,8	0,1,2,4,5,6,7,9,10,11,12,13
Data4		0,1,2,3,4,5,6,7,8
Data5		0,1,2,3,4
Data6	11	0,1,2,3,4,5,6,7,8,9,10
Data7	0,1,2,4	3,5,6

閥值=20	類別型特徵(Column)	連續型特徵(Column)
Data1	7	0,1,2,3,4,5,6,8
Data2	0,1,2,3,4,5,6,7	8
Data3	3,8	0,1,2,4,5,6,7,9,10,11,12,13
Data4		0,1,2,3,4,5,6,7,8
Data5		0,1,2,3,4
Data6	11	0,1,2,3,4,5,6,7,8,9,10
Data7	0,1,2,3,4	5,6

2.3 使用相關性填補遺失值

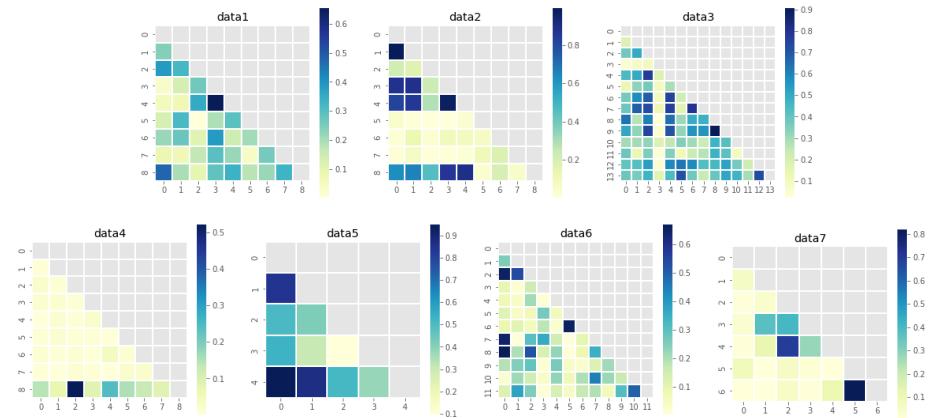
2.3.1 各個特徵與 Y 的相關性

從下圖熱圖可以發現，各個訓練集中，有些特徵與 Y 具有高相關性，因此我們希望利用此因素使填補遺失值更加準確，而新增一項門檻讓那些有較高相關係數的變數放在 Regressor 預測。



2.3.2 各個特徵之間的相關性

我們將相關性取絕對值（如下圖），可以發現有些特徵除了與 Y 具高相關，特徵與特徵之間也有相關性，可以納為考量因素之一，因此更進一步將門檻改為若此特徵不是與 Y 具有最高相關性，則改為與此特徵最高相關的特徵變數。



2.4 填補遺失值

我們嘗試以下方式將上述轉換後的資料進行遺失值填補。

2.4.1 KNNImputer

超參數：`n_neighbors=5, weights=uniform`。

缺點：僅使用此方法無法對類別型變數進行分類預測，因此後續無多做嘗試。

2.4.2 IterativeImputer

為了彌補 KNNImputer 的缺失，嘗試 IterativeImputer，對類別變數與連續變數使用不同的模型進行預測。

◆ KNN

模型：類別變數為 KNeighborsClassifier，連續則為 KNeighborsRegressor。

超參數設定如下表：

	n_neighbors	weights	leaf_size
KNeighborsClassifier	15	distance	50
KNeighborsRegressor	15	distance	-

◆ Xgboost

模型：類別變數為 XGBClassifier，連續變數為 XGBRegressor。

超參數設定如下表：

	n_estimators	max_depth	eval_metric	objective
XGBClassifier	-	5	[logloss, auc, error]	binary:logistic
XGBRegressor	100	-	-	-

2.5 模型預測

填補遺失值完成後，再使用 XGBoost 進行預測。若訓練資料的 Y 的資料種類少於 20，則使用 XGBClassifier，大於 20 則使用 XGBRegressor。

超參數設定如下表：

	n_estimators	max_depth	eval_metric	objective
XGBClassifier	-	5	[logloss, auc, error]	binary:logistic
XGBRegressor	100	-	-	-

3. Experimental analysis

以下方法在遺失值填補表現的觀察為將資料集各個欄位的遺失值填補後，取各個解釋變數 x 與應變數 y 繪製散佈圖，觀察遺失值的填補情形。其中非遺失值的資料點以藍色表示，遺失值經過填補後的資料點用紅色表示。

3.1 方法一

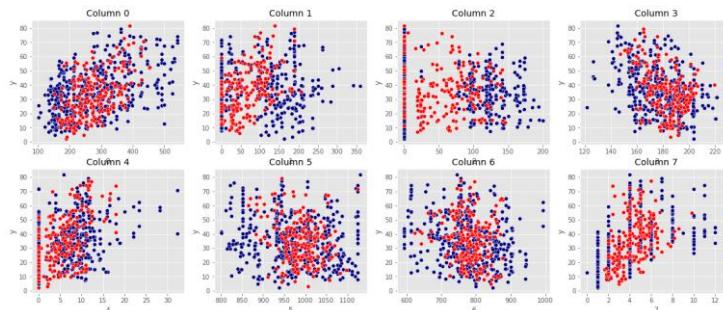
3.1.1 方法及結果

從前面各個資料集中各個特徵對 Y 的關係散佈圖可以看出許多變數呈現直向的，因此我們將這些變數當作類別變數。且再進行遺失值填補時設置當類別變數的水準數 ≤ 10 時便使用使用 KNNImputer 填補。

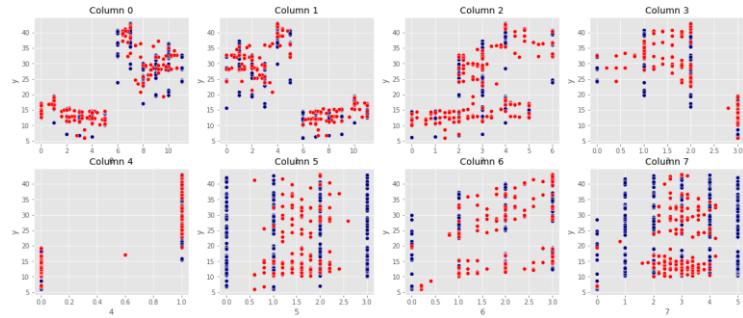
遺失值填補		方法	KNNImputer			
		超參數	n_neighbors=5, weights=uniform			
模型預測	變數水準 ≤ 10	方法	XGBClassifier			
		超參數	max_depth=5, eval_metric=['logloss','auc','error'], objective='binary:logistic', use_label_encoder=False			
預測模型 MAE	變數水準 > 10	方法	XGBRegressor			
		超參數	n_estimators=100			
預測模型 MAE		Data1	Data2	Data3	Data4	
		61.08	1.77	25.38	0.03	
上傳結果		Data5	Data6	Data7		
		14.8	0.45	3.02		
上傳結果		MP_mean	221.3422			
		TP_mean	34.2522			

3.1.2 遺失值填補表現

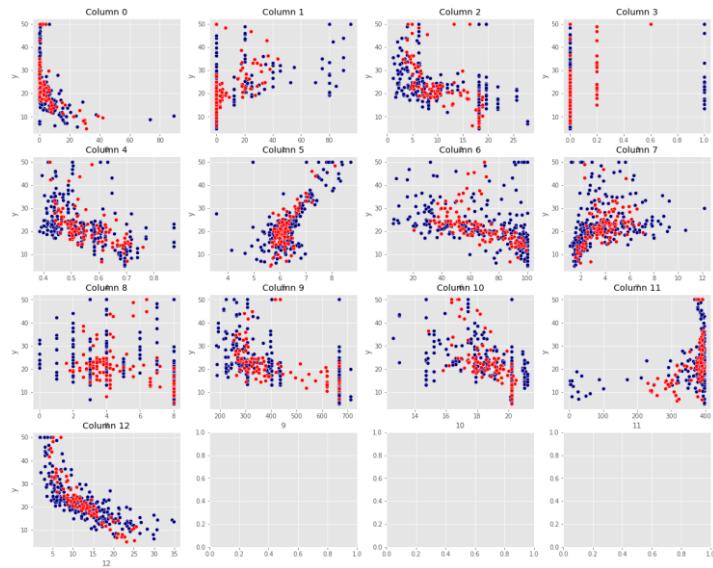
◆ Data1



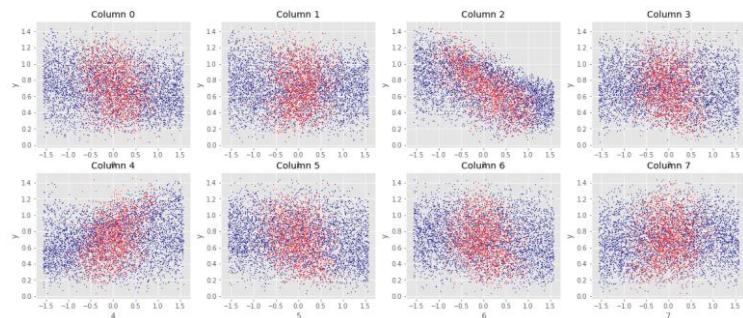
◆ Data2



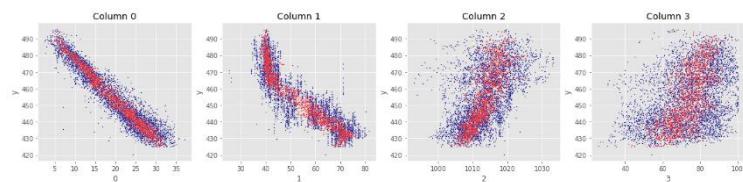
◆ Data3



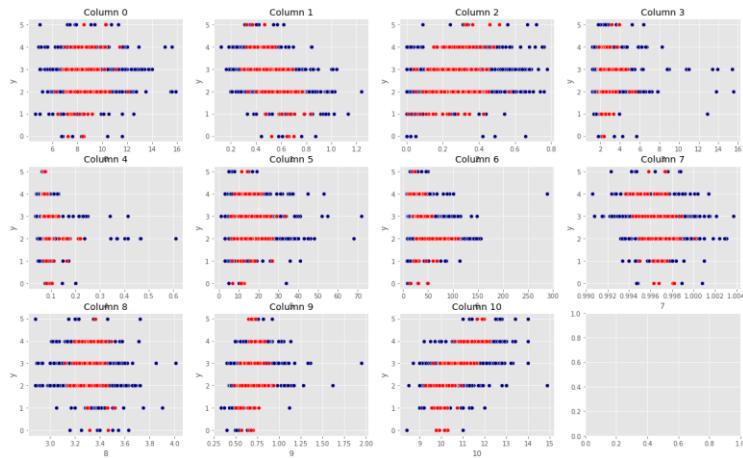
◆ Data4



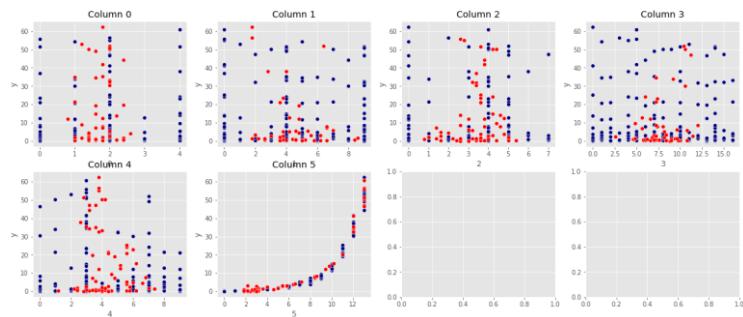
◆ Data5



◆ Data6



◆ Data7

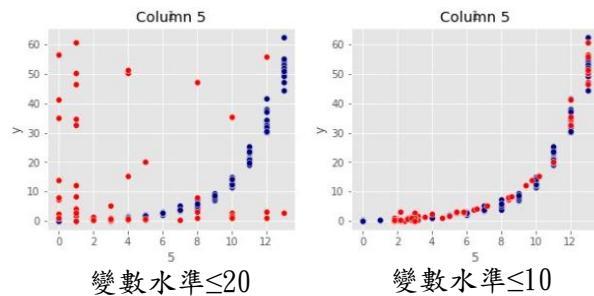


3.1.3 結果陳述：

原先設置當類別變數的水準數 ≤ 20 時便使用 KNNImputer 填補，但以 Data7 的 Column 5 為例，發現類別變數的水準數 ≤ 20 時的遺失值填補效果不好(如下圖)，因此改為設置當類別變數的水準數 ≤ 10 時便使用 KNNImputer 填補。

對於遺失值進行填補的表現，可以發現 KNNImputer 對類別變數的遺失值填補效果並不佳，以 Data7 的 Column 0,1,2,3,4 為例，多數填補的遺失值非已有的數值。由於僅使用此方法無法對類別型變數進行分類預測，所以下我們嘗試將類別與連續變數分開各使用 Classifier 與 Regressor 進行 NA 值預測。

此外，可以發現 data7 的 Column 5 與 y 值有很大相關，但因為 y 值不是類別變數，所以如果我們將該變數分做類別變數會導致在 KNNImputer 的模型下沒辦法用 y 值去補該類別變數的遺失值，所以我們接下去觀察變數與 y 值的相關係數來進行彌補。



3.2 方法二

3.2.1 方法及結果

使用 IterativeImputer 配合模型，將連續及有序資料變數區分開來進行預測，並將變數間相關係數 >0.8 的變數一起放入 KNeighborsRegressor 對遺失值進行填補。

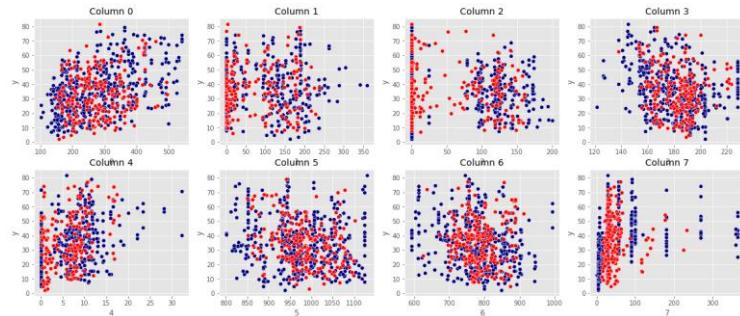
此外，模型填值的順序為：

- (1.) 判斷連續變數多，還是類別變數多
- (2.) 若連續變數多於類別變數，先將全部的連續變數以及 y 值建立 Regressor 模型填補連續變數的缺失值，再分別對每個類別變數的缺失值用 Classifier 模型填補
- (3.) 若類別變數多於連續變數，先把全部的類別變數建立 Classifier 模型填補類別變數的遺失值，再把整個資料集放進 Regressor 模型對連續變數的缺失值進行填補

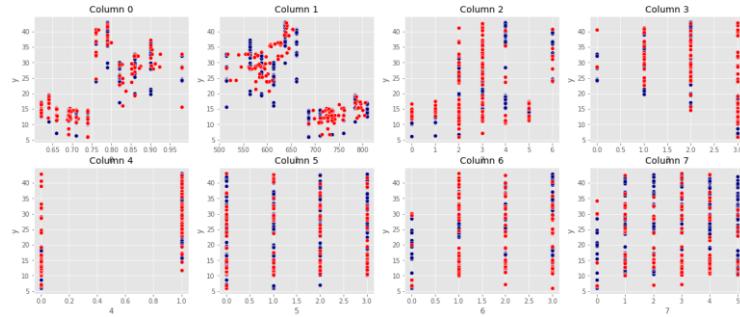
遺失值 填補	類別變數填補	方法	KNeighborsRegressor	
		超參數	n_neighbors=15, weights='distance'	
	連續變數填補	方法	KNeighborsClassifier	
		超參數	n_neighbors=15, weights='distance', leaf_size=50	
模型預測	變數水準數 ≤ 10	方法	XGBClassifier	
		超參數	max_depth=5, eval_metric=['logloss','auc','error'], objective='binary:logistic', use_label_encoder=False	
	變數水準數 > 10	方法	XGBRegressor	
		超參數	n_estimators=100	
預測模型 MAE	Data1	Data2	Data3	Data4
	84.05	5.10	31.21	0.041
	Data5	Data6	Data7	
	11.91	0.58	4.79	
上傳結果	MP_mean	154.0797		
	TP_mean	31.6734		

3.2.2 遺失值填補表現

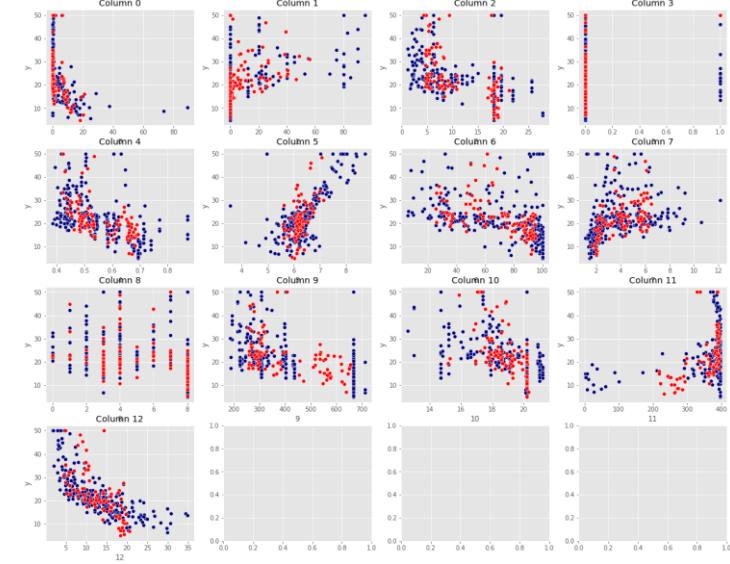
◆ Data1



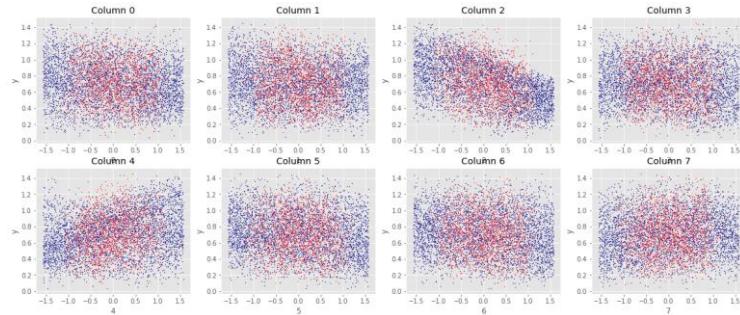
◆ Data2



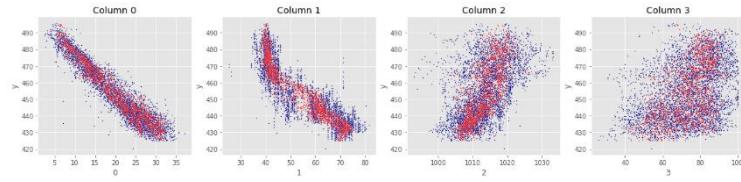
◆ Data3



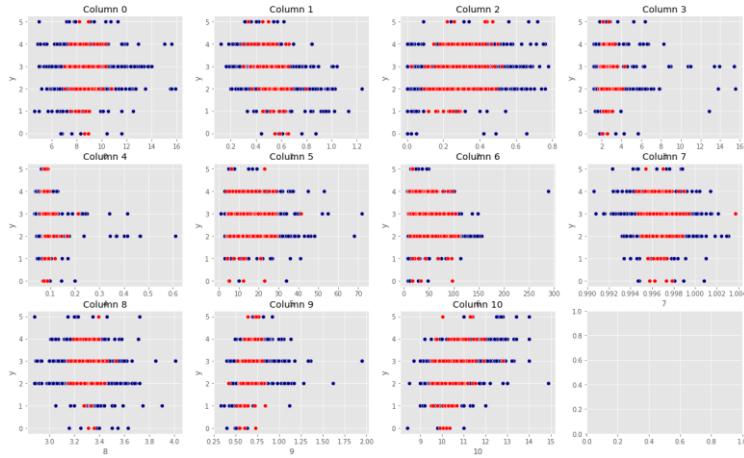
◆ Data4



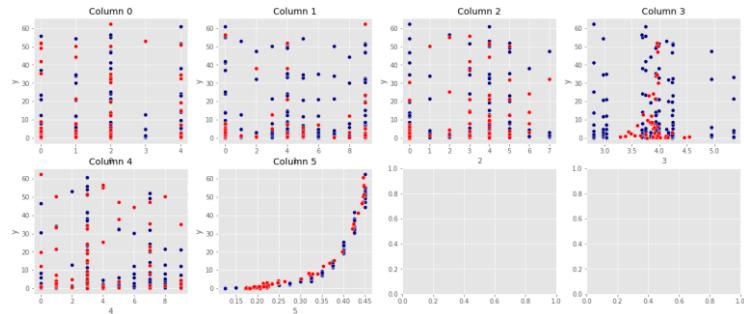
◆ Data5



◆ Data6



◆ Data7



3.2.3 結果陳述：

在這次的遺失值填補的方法中，我們將變數間相關係數 >0.8 的變數一起放入 KNeighborsRegressor 對遺失值進行填補，期望提高最終模型的準確度。那從最終的模型結果可以看到 data7 的 MAE 明顯下降，但 data1 却大幅提升，data3 小幅上升。因此我們往回看點散圖可以發現因為 data3 的 Column 3 經過相關係數篩選被移動至 Regressor 模型中，導致與其更相關的 Column 1,2 估計的能力降低。因此接下去我們改成挑選與 y 值有高相關的變數才移動至 Regressor 模型去對遺失值進行填補。

從上傳結果來看的話，使用 IterativeImputer 配合 KNN 模型對遺失值進行填補可以明顯看出對於類別變數的填補較為合理，填補的結果也較方法一好很多。

3.3 方法三

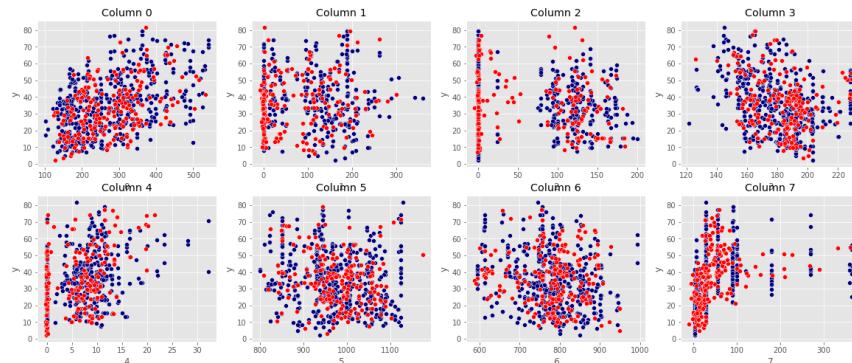
3.3.1 方法及結果

使用 IterativeImputer 配合模型，將連續及有序資料變數區分開來進行預測，並挑選與 y 值有高相關的變數才移動至 Regressor 模型去對遺失值進行填補。
(模型填值的順序同方法二)

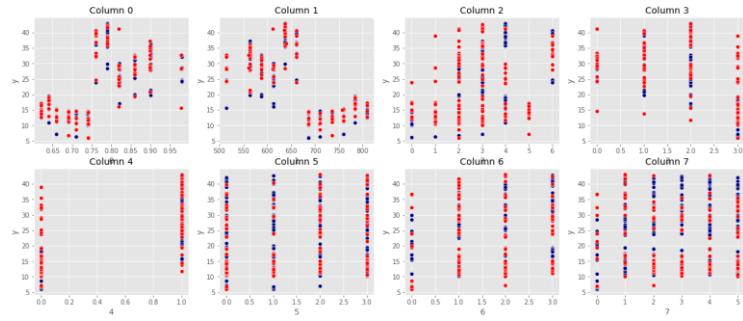
遺失值 填補	類別變數填補	方法	XGBRegressor
		超參數	n_estimators=100
連續變數填補		方法	XGBClassifier
		超參數	max_depth=5, eval_metric=['logloss','auc','error'], objective='binary:logistic', use_label_encoder=False
模型預測	變數水準數 ≤ 10	方法	XGBClassifier
		超參數	max_depth=5, eval_metric=['logloss', 'auc', 'error'], objective='binary:logistic', use_label_encoder=False
	變數水準數 > 10	方法	XGBRegressor
		超參數	n_estimators=100
預測模型 MAE		Data1	Data2
		33.74	5.07
		Data5	Data6
		12.72	0.49
上傳結果		Data3	Data4
		13.90	0.02
		Data7	
		1.35	
		MP_mean	151.7618
		TP_mean	30.3960

3.3.2 遺失值填補表現

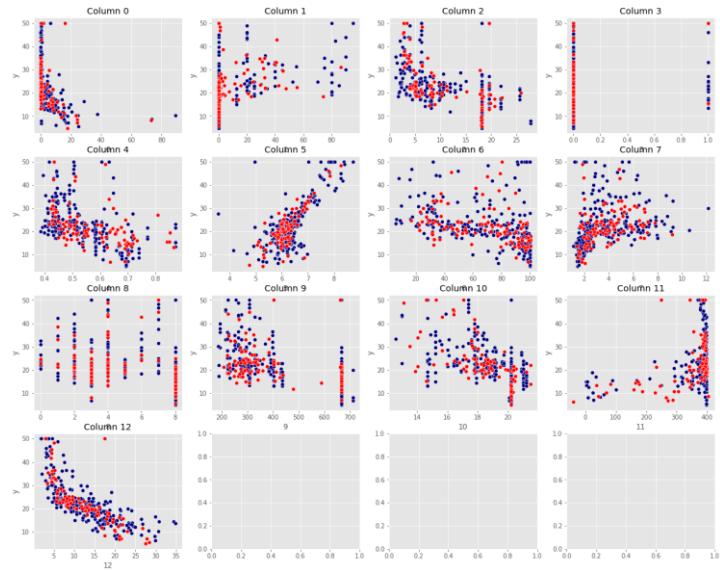
- Data1



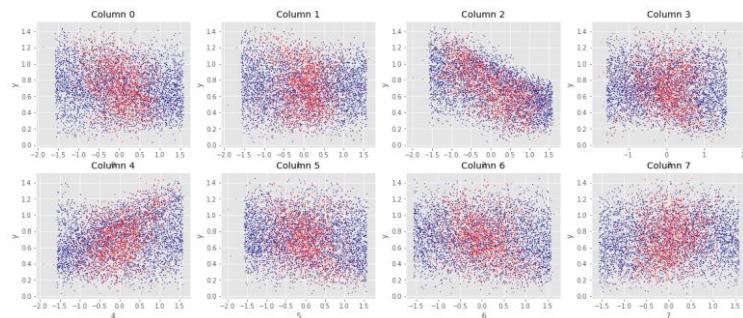
◆ Data2



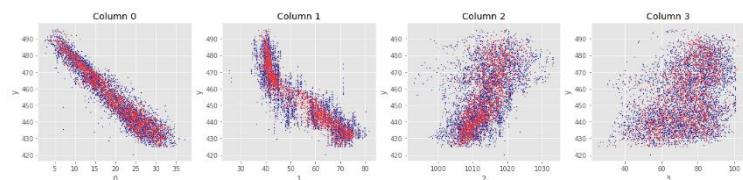
◆ Data3



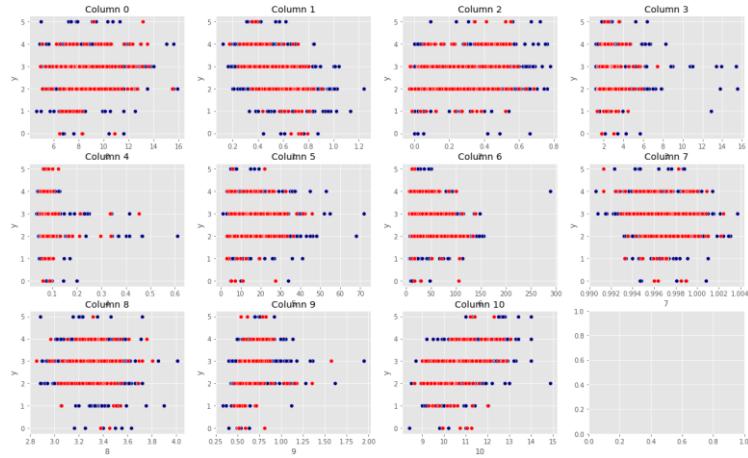
◆ Data4



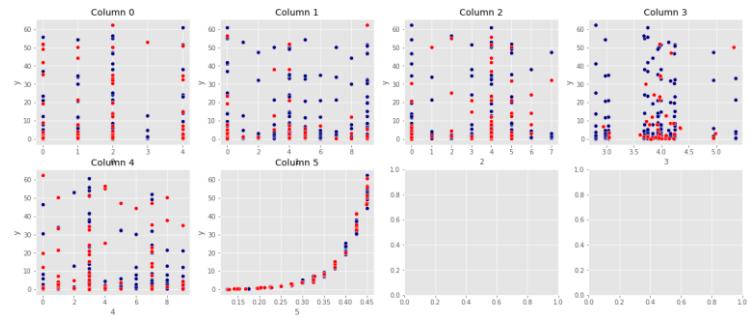
◆ Data5



◆ Data6



◆ Data7



3.3.3 結果陳述：

這次遺失值的填補改用 IterativeImputer 配合 Xgboost 的模型填補，填補的 MAE 比起使用 KNN 模型下降了一點，但從排行榜上觀察發現排行前三在填遺失值上 MP_mean 的分數都在 100 以下，所以在填遺失值的部分還能再改進！

3.4 方法四

3.4.1 方法及結果

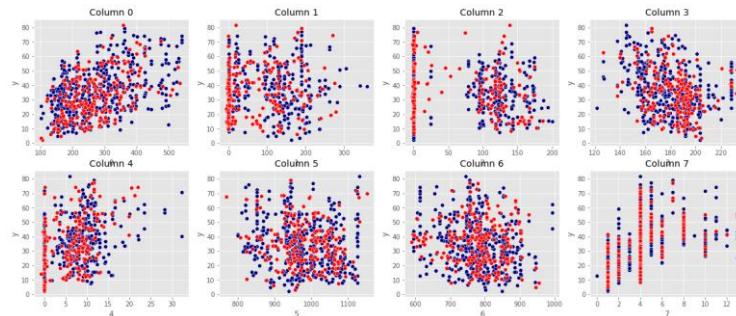
使用 IterativeImputer 配合模型，將連續及有序資料變數區分開來進行預測，但 IterativeImputer 在使用 Classifier 時無法套用連續變數。

(模型填值的順序同方法二)

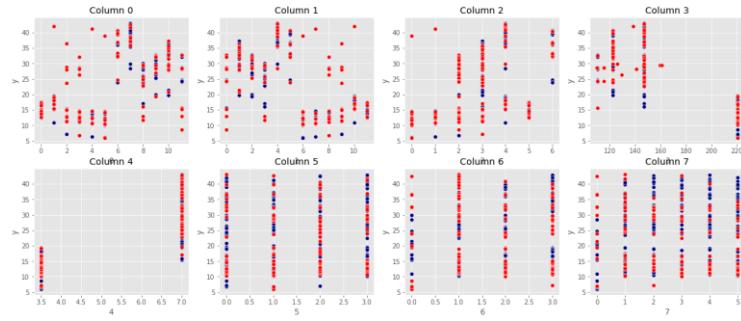
遺失值 填補	類別變數填補	方法	XGBRegressor
		超參數	n_estimators=100
連續變數填補	方法	XGBClassifier	
		超參數	max_depth=5, eval_metric=['logloss','auc','error'], objective='binary:logistic', use_label_encoder=False
模型預測	變數水準數 ≤ 10	方法	XGBClassifier
		超參數	max_depth=5, eval_metric=['logloss','auc','error'], objective='binary:logistic', use_label_encoder=False
	變數水準數 > 10	方法	XGBRegressor
		超參數	n_estimators=100
預測模型 MAE	Data1	Data2	Data3
	59.55	7.68	6.10
	Data5	Data6	Data7
	13.81	0.51	2.73
上傳結果	MP_mean	151.7696	
	TP_mean	30.2521	

3.4.2 遺失值填補表現

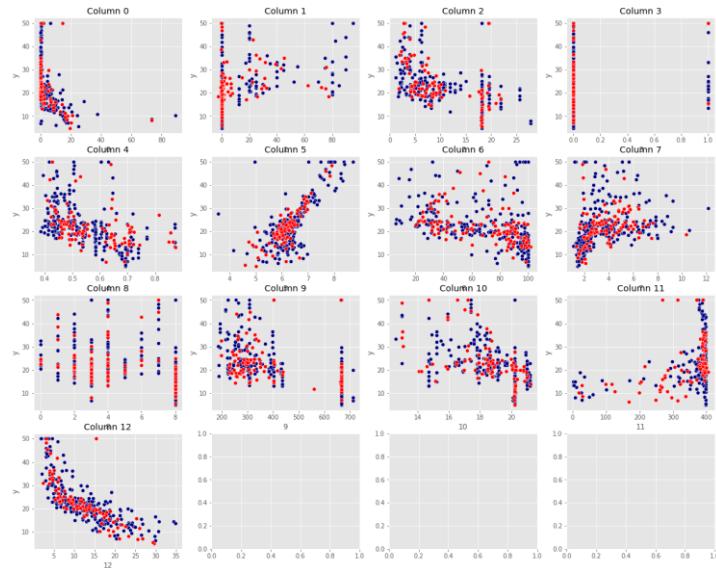
- Data1



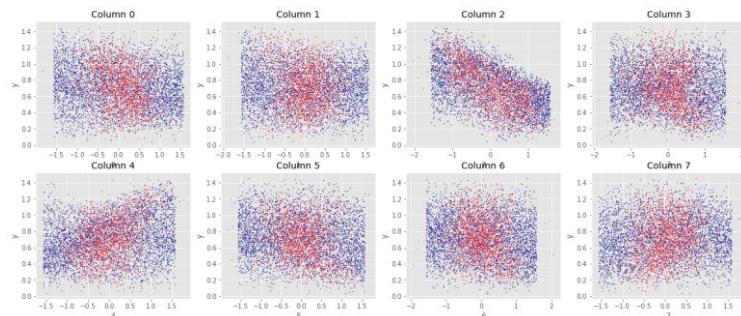
◆ Data2



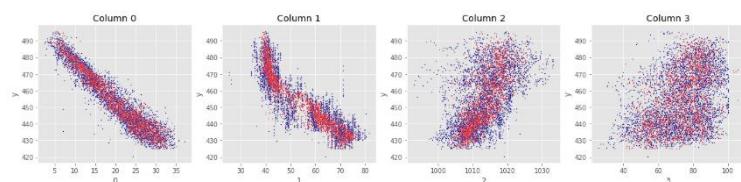
◆ Data3



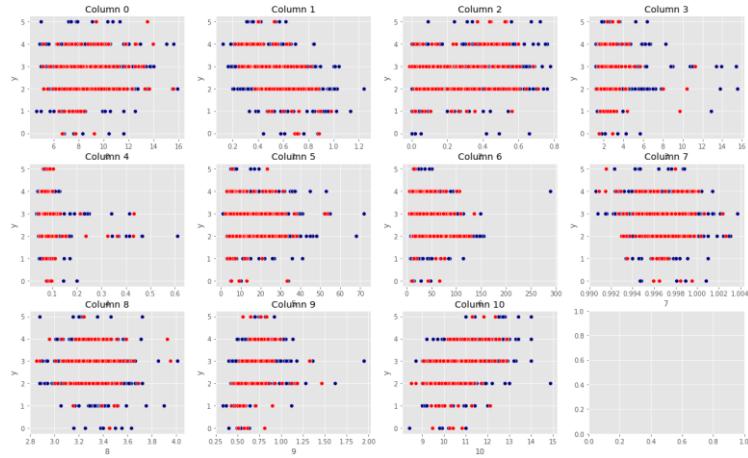
◆ Data4



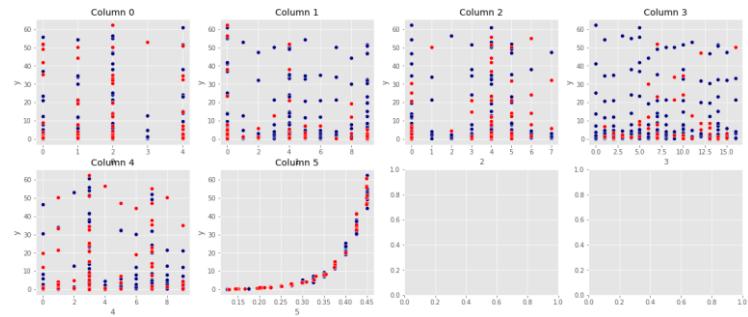
◆ Data5



◆ Data6



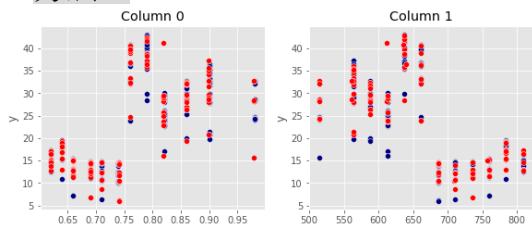
◆ Data7



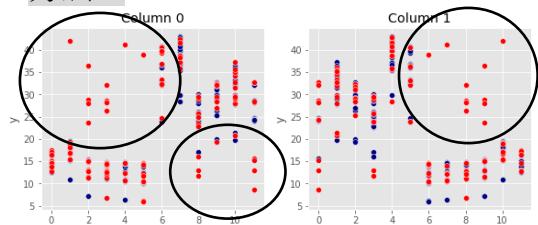
3.4.3 結果陳述：

若觀察遺失值的填補表現，可以發現在 Data2 中 Column 0,1 的填補較方法三差，多出了許多並非已知資料的數值(如下圖)，所以希望可以藉由調整模型參數等方法來優化遺失值的填補。

方法三 Data2



方法四 Data2



3.5 方法五

3.5.1 方法及結果

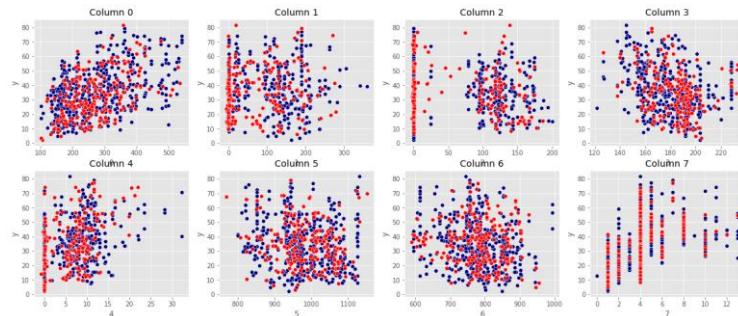
使用 IterativeImputer 配合模型，將連續及有序資料變數區分開來進行預測，但 IterativeImputer 在使用 Classifier 時無法套用連續變數。

(模型填值的順序同方法二)

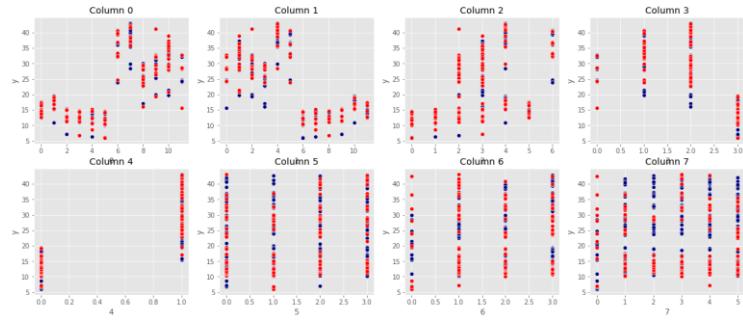
遺失值 填補	類別變數填補	方法	XGBRegressor
		超參數	n_estimators=100
連續變數填補		方法	XGBClassifier
		超參數	max_depth=6, eval_metric=['logloss', 'auc', 'error'], objective='binary:logistic', use_label_encoder=False
模型預測	變數水準數 ≤ 10	方法	XGBClassifier
		超參數	max_depth=6, eval_metric=['logloss', 'auc', 'error'], objective='binary:logistic', use_label_encoder=False
	變數水準數 > 10	方法	XGBRegressor
		超參數	n_estimators=100
預測模型 MAE		Data1	Data2
		34.96	8.28
		Data5	Data6
		12.06	0.44
上傳結果		Data3	Data4
		7.58	0.02
		Data7	
		0.52	
上傳結果	MP_mean	151.7666	
	TP_mean	30.2530	

3.5.2 遺失值填補表現

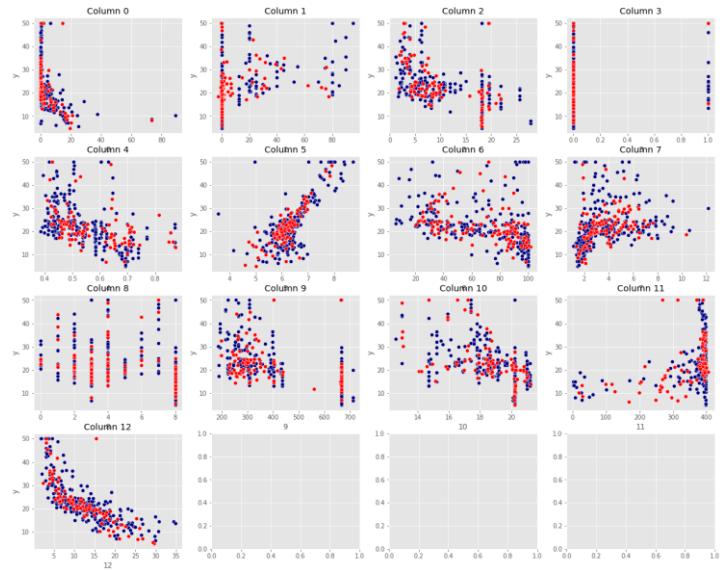
- Data1



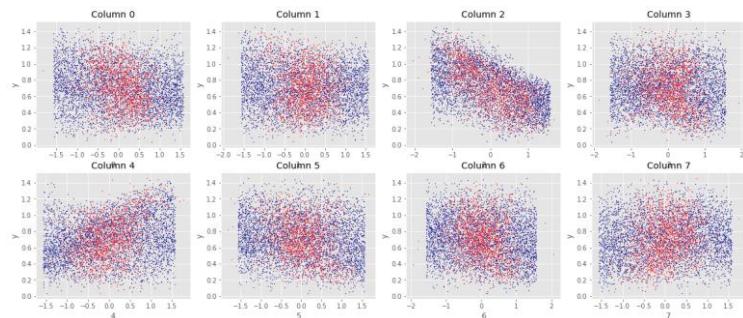
◆ Data2



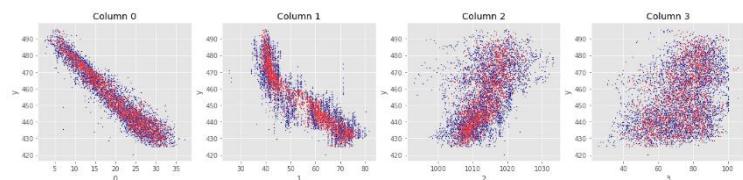
◆ Data3



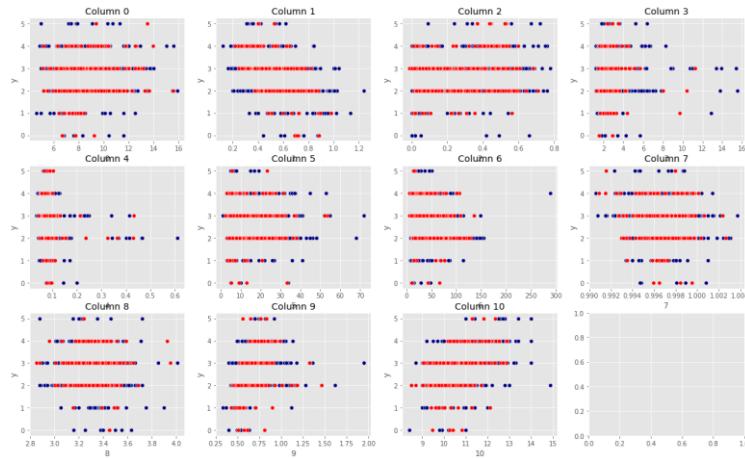
◆ Data4



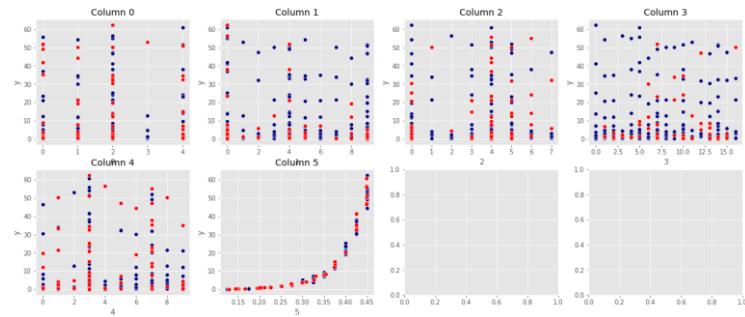
◆ Data5



◆ Data6



◆ Data7



3.5.3 結果陳述：

對於填補遺失值上的 Xgboost 模型的超參數進行調整，將 `max_depth` 另為 6，增加模型的層數，希望可以得到比較好的結果，但是依然沒有明顯的進步。

3.6 方法比較

這次的競賽我們用於填補遺失值的方法主要著重在遺失值得填補，並利用點散圖去觀察填補的情形是否可以進行改進，使用的模型包括 KNNImputer、KNeighborsRegressor、KNeighborsClassifier、XGBRegressor、XGBClassifier，其中 Xgboost 的表現皆較好。模型預測的部分僅使用 Xgboost 中的 XGBRegressor、XGBClassifier 進行配適。總結而言，可以發現用 Xgboost 進行遺失值填補以及模型配適的預測結果較好，但在超參數上的調整卻無法得到大幅度的進步。

在使用 IterativeImputer 配合 KNN 模型來填補遺失值時，或許是因為 KNN 模型的較單一，對於遺失值填補的較不精確；而配合 Xgboost 模型填補遺失值，雖然有嘗試調整超參數，但是填補的表現也沒有明顯的進步，有可能是因為過度配適，因為在我們自己檢測時 y 值的 MSE 是有降低的，但是上傳後的預測排行榜上卻沒有進步。

4. Conclusion

最初拿到資料時，使用單一方法進行填補，使得類別變數沒有太好的預測，而後尋找方法對於變數進行分類，加以區分成 Regressor 與 Classifier 進行遺失值填補。雖然最後改進相關係數方法的時候並沒有得到太多的進步，但納入相關係數為考量的前後，有很大的結果差別，也就是說填補缺失值的時候變數間的相關性非常重要，但因為這次的競賽必須使用 single model，我們沒有辦法對每個資料集進行詳細的調查與調整，導致改進每一個方法時需要犧牲的條件都不同。最後可惜的部分是無法對於資料集內不平衡的欄位進行彌補，Data 1 跟 Data 3 都有些變數包含許多的 0，若能做到這點，也許能更好的填補 Data 1。

另外，因為我們所嘗試的遺失值填補方式為將類別及連續變數分開討論，因此若類別與連續變數間有顯著的相關性存在就會被我們忽略，或許這個部分是我們在未來的競賽可以多考慮的部分。

5. Citation

- (1) 分析工具箱：K 近邻填补缺失值
<https://reurl.cc/82DZ9b>
- (2) Sklearn - Imputation of missing values
<https://reurl.cc/dxQjak>
- (3) Sklearn - sklearn.impute.IterativeImputer
<https://reurl.cc/95DYbv>
- (4) Sklearn - sklearn.impute.KNNImputer
<https://reurl.cc/2oKk74>