簡易快速的資料前處理 - 於PostgreSQL

一、介紹

隨著硬體的進步與知識的累積，人們對於資料有了越來越多的應用。這些應用帶來為生活帶來便利，也為公司們帶來龐大的商業利益。很多人將資料稱為新時代的石油，各大企業也紛紛循著各種管道取得資料。此外，隨著網路的普及，資料的取得也愈加方便。然而面對如此龐大筆數與眾多欄位的原始資料，我們如何從中擷取有用的訊息勢必成為關鍵。此外無論是對於分散式資料庫的使用者或是需遠端傳輸資訊的跨國公司而言，能在資料庫端進行前處理在進行資料傳輸都能省下需多的時間成本，建置資料庫的超級電腦也能妥善發揮其性能。我們的專案使用 Plpy 套件[1] 在 PostgreSQL [2] 上實作了能在資料庫進行前處理的功能，指令十分簡便。在填補缺失值、變數選擇和極端值偵測等前處理都僅需一行指令即可解決，而且能自動辨別、處理各種資料型態，無需多花功夫對資料進行瞭解，並且以有效率運用記憶體的方式回傳結果。

二、方法

- Missing Values

缺失值填補的方法可以分為兩類，一類是 Univariate Imputation，例如直接該欄填入中位數，類別型變數則填入出現頻率最高的類別。另一類則為 Multivariate Imputation ，利用其他欄位所提供的資訊來判斷適當的填補值。前者的做法簡單快速，但由於我們實做前處理是為了提升後續機器學習任務表現，因此我們會採用後者的策略，來達到較精確的填補。步驟如圖（一）。

Algorithm

Let $X_1, X_2, \dots, X_p$ be columns that contains missing values. (sorted by #missing values in ascending order)
For all $X_i, i = 1,2, \dots, p$:

1. *Create model M_i. M_i is a regressor if X_i is numerical else a classifier.*
2. *Take all columns except X_i as input, X_i as output. Take all the rows where X_i has no missing value as the training data to train M_i.*
3. *After training, take all columns except X_i as input. Take all the rows where X_i has missing value as the testing data to feed into M_i.*
4. *The result is the prediction of missing-values in X_i. Fill them to X_i.*

圖（一）

對於上述迴歸與分類模型，我們考慮以下兩點來決定使用的機器學習演算法：

1. 由於在填補欄位 Xi 時，其餘的欄位亦會有缺失值，且同時會有數值型和類別型的欄位，因此我們考量使用同時可以處理缺失值和類別型輸入的模型。

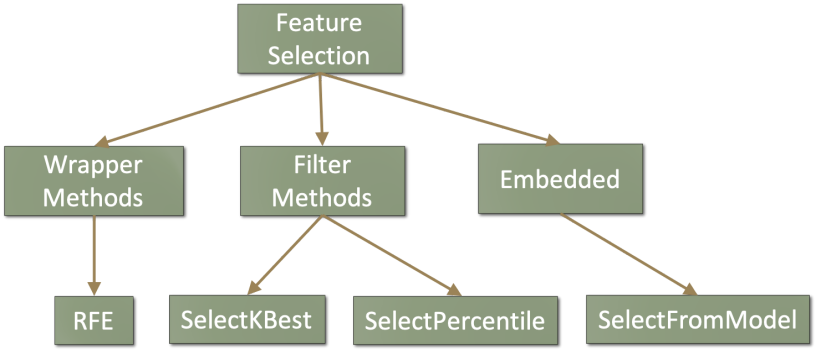
2. 由於對於每個含有缺失值的欄位都需建模，時間複雜度較高，因此也不該採用太花計算時間的模型。

綜和兩點考量，我們會使用 Tree-based gradient boosting 作為預測模型。

- Feature Selection

在 Feature Selection 的部分，我們依照方法的不同將 selection 分成三個部分，如附圖。分別是 Wrapper Methods、Filter Methods 以及 Embedded。其中 Filter Methods 中又包含了兩種透過統計分析 feature 與 target 相關性的方法。以下我們將依序介紹每一種 selection。

  - RFE：提供不同的 estimator，讓使用者依據 model 的類型做選擇。將所有 feature 經過多次 iteration 後篩選出最符合的 feature。
  - SelectKBest：讓使用者透過輸入一個 K 值來決定最終要選出來的 feature 有幾個。
  - SelectPercentile：讓使用者藉由輸入 threshold 當作基準，將大於 threshold 的 feature 選出。
  - SelectFromModel：提供不同的 estimator，讓使用者依據 model 的類型做選擇。並且不同於 RFE ，此方法將所有 feature 經過 estimator 後一次性的將適合的 feature 選出來。



圖(二)

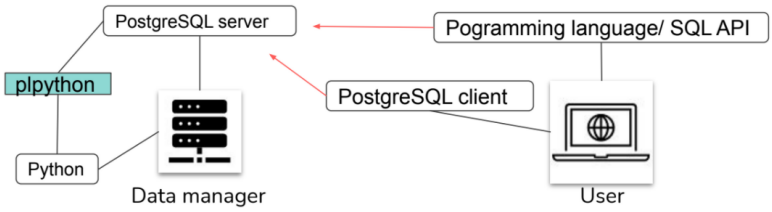
- Outlier Detection

  在蒐集完筆數眾多的資料後，往往會遇到諸如量測錯誤或填寫錯誤等問題，導致所蒐集的資料集有單變量或多變量的極端值。這種來自不同抽樣機制的極端值會影響諸如平均數、相關係數、最小方差回歸等統計推論。因此資料使用者往往需要在前處理時找出我們懷疑為極端值的樣本，以利估計出更符合我們愈推論樣本的模型。由於蒐集資料的便利性增加，現在的資料動輒有數十萬筆資料、數萬個變數，我們參考了 Santoyo [4] 和 Xu et al. [5] 找到了四種在高維度也能有不錯表現的異常值偵測方法，並使用 sci-kit learn 的套件 [6] 將其嫁接在 PostgresSQL 平台中使用。

  - One Class SVM： 透過 RBF kernel 的 SVM 演算法根據樣本點的特徵去訓練出一個邊界，並以這個邊界將資料集區分為正常資料和異常資料 [7]。
  - IsolationForest：首先將樣本點擬和二元決策樹模型，再將樣本點少且與其他樣本點少的分枝視為極端值。
  - DBSCAN：透過樣本點周圍一定距離內可達到的其他樣本數量多寡來分群，將樣本點密集的區域分為正常資料，周圍樣本數少或是無其他樣本者分為異常資料 [8]。
  - FAST-Minimum Covariance Determinant (FAST-MCD)： 計算樣本點的共變異數矩陣，假設樣本服從高維度高斯分配，並抽出離群值。

三、實作

我們實作的框架如下頁圖（三），我們會在資料端建立 PostgreSQL server ，並安裝其Python 延伸套件。使用者可以在 PostgresSQL 提供的使用者介面，或是透過其他程式語言的資料庫 API 來操作我們所實現的函數。當 PostgreSQL 收到指令時，會呼叫 Python runtime，Python 則透過 plpy 介面讀取資料庫內的指定 TABLE ，完成處理再寫回資料庫。在我們的實作中，除了 Missing value 會直接寫回資料庫，其餘兩個函數皆可以藉由不佔額外記憶的方式讓使用者可以取用篩選過後的資料。



圖（三）

實務上，使用基於機器學習的前處理方法時，必須先分開數值型和類別型的欄位，並作相對應的處理後再結合，為了免除使用者的困擾，我們提供的函數皆會自動判斷 TABLE 的欄位為數值或類別型，並做好處理，因此使用上非常簡單，只要指定欲處理的表格即可。以下介紹各函數的使用方法：

共同參數：

參數名稱	資料型別	意義
table_name	varchar	輸入的 table 名稱。
save_as	varchar	輸出的 table 名稱（feature selection則是create view）。
target	varchar	target 的 column名稱。

表（一）共同參數

- Missing Values

  補值方法已詳述，我們採用 LightGBM[9] 中 LGBMRegressor 實作其中迴歸模型、LGBMClassifier 實作分類模型。

  - FillMissing

    SELECT FillMissing(table_name, save_as, target);

- Feature Selection

  在 Feature Selection 的部分，我們參考了 Scikit-learn [10] 中的四個 Feature Selection [3] 的方法，並且基於該函式的參數實做了四個 function 供使用者使用，以下分別介紹語法，並整理參數的資料型別與代表意義於表（二）。

- RFE

  SELECT RFE (table_name, save_as, target, estimator_name, n_features_to_select, step);

- SelectKBest

  SELECT SelectKBest (table_name, save_as, target, score_func, k);

- SelectPercentile

  SELECT SelectPercentile (table_name, save_as, target, cat_names, percentile);

- SelectFromModel

  SELECT SelectFromModel (table_name, save_as, target, estimator_name, threshold, max_features);

參數名稱	資料型別	意義
estimator_name	varchar	用來估量 feature 重要性的 estimator 名稱。
n_features_to_select	real	要選擇的 feature 數。預設為 null，會選擇一半。如果是整數，即代表要選擇的個數；如果是在 0 和 1 之間的浮點數，則代表要選擇的比例。
step	real	每次 iteration 要移除掉的 feature 數。預設為 1。如果是整數，即代表要每次要移除個數；如果是在 0 和 1 之間的浮點數，則代表每次要移除的比例。
score_func	varchar	用來估量 feature score 的 function 名稱。有 f_regression, f_classif, chi2, mutual_info_regression, mutual_info_classif。
k	int	要選擇的 feature 個數，預設為 10。
cat_names	varchar	指定是 categorical feature 的 columns名稱。
percentile	int	要保留的 feature 數量百分比。
threshold	varchar	Feature 重要性的閾值。
max_features	int	最多要選擇的 feature 數量。

表（二）Feature Selection 參數說明

- Outlier Detection

  我們分別採用 Scikit-learn中的 svm, covariance, ensemble 和 cluster 套件來達成四種不同的偵測方法，使用者可以在參數中選擇想要使用的方法，見表(三)。

  - Outlier Detection

    SELECT OutlierDetection(table_name, save_as, method,

    target_name, index_column);

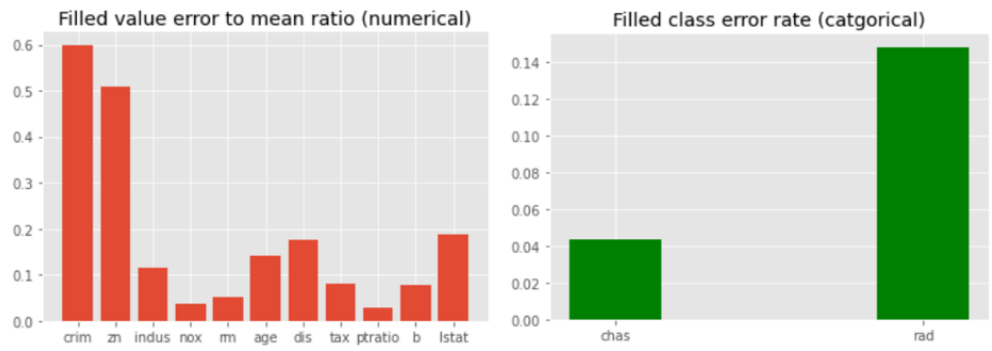
參數名稱	資料型別	意義
method	varchar	指定用來偵測極端值的演算法，包含 "OCSVM", "IsolationForest", "DBSCAN", 和 "COV"
index_column	varchar	可以指定primary key的欄位，極端值資料會根據該欄回傳。若無指定則會創建新的欄位以記錄是否為極端值。

表（三） Outlier Detection 參數說明

四、實驗

我們以 Boston 資料集示範，將一個有缺失的 TABLE 補滿、挑選出對預測 price（房價）較有幫助的變數、剔出異常值，最後以預測房價的任務證明前處理的幫助。實驗中我們會將欄位 chas 及 rad 以類別型變數來操作，並隨機在 25% 資料中，故意丟失一到三個欄位的資料，來模擬現實狀況中有缺失的資料。

經過 FillMissing 後，會得到一完整 TABLE，由於缺失值是我們故意丟失，因此我們可以評估預測值和真實值的差異（圖四），左圖為數值型變數的 L1 誤差佔變數平均比例，右圖為類別型變數的預測錯誤率。

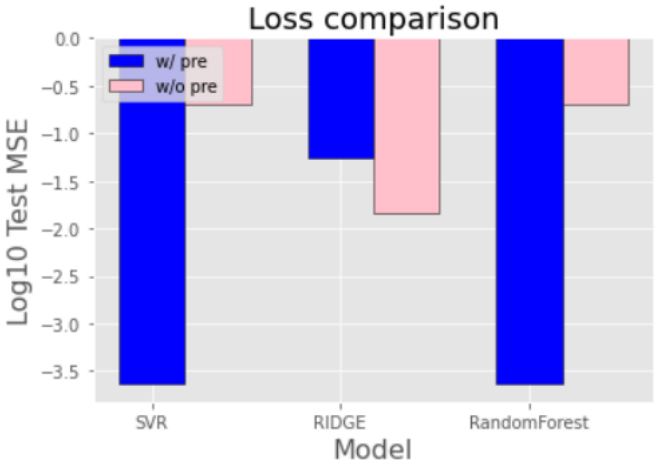


圖（四）

經過 RFE 之後（使用GradientBoostingRegressor），保留下列有助於預測房價的變數：crim, nox, rm, dis, ptratio, lstat

經過 OutlierDetection（使用COV），我們挑出 41 筆 outliers（rows）並將之移除。

我們將經過前處理的資料和原始資料（移除有缺失的 row，佔25%），分別使用三種迴歸模型對房價作預測，並以新的 100 筆資料作測試，結果如圖（五）。



圖(五)

可以發現在使用 SVR、Randomforest 等非線性模型時，經過前處理的資料使房價預測錯誤下降近 1000 倍，在 ridge regression 則會提升數倍，總體而言經前處理可以得到更低的誤差。

五、結論

實作上我們發現有使用資料前處理對於模型的預測表現會提升許多，在線性模型表現較差，是因為特徵選取是透過非線性演算法進行，若使用者想用線性模型來做預測，也可以改用線性的特徵選取。總結來說本專案透過 Python 跟 SQL 的串聯達成在資料庫進行資料前處理，這對日益增大的數據集和需要遠端傳輸的資料都能有所幫助，使用者只需撈回和自身任務相關的資料，節省網路I/O。

參考資料

- [1] PL/Python - Python Procedural Language
- [2] Postgresql Documentation
- [3] scikit learn-feature selection
- [4] Sergio Santoyo (2017). A Brief Overview of Outlier Detection. Retrieved 2021/6/27.
- [5] Youngmi huang (2019). OneClass SVM：異常檢測任務 (Anomaly Detection) 的算法理解與實踐. Retrieved 2021/6/27.
- [6] scikit learn-outlier detection
- [7] Kavya Gajjar (2020). Cluster Analysis with DBSCAN : Density-based spatial clustering of applications with noise. Retrieved 2021/6/27.
- [8] Xiaodan Xu, Huawen Liu, Li Li, Minghai Yao (2018). A Comparison of Outlier Detection Techniques for High-Dimensional Data. International Journal of Computational Intelligence Systems, Volume 11, Issue 1, pp.652-662.
- [9] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, Tie-Yan Liu. "LightGBM: A Highly Efficient Gradient Boosting Decision Tree". Advances in Neural Information Processing Systems 30 (NIPS 2017), pp. 3149-3157
- [10] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

分工表

姓名	學號	貢獻
曾尹均	R09922080	1. Feature Selection文獻回顧 2. Feature Selection 程式實作 3. Mac OS 環境建置 4. 實驗設計
張烱郁	R09946031	1. Missing Value Imputation文獻回顧 2. Missing Value Imputation 程式實作 3. Windows 環境建置 4. 實驗設計
林祐辰	R08323046	1. Outlier Detection 文獻回顧 2. Outlier Detection 程式實作 3. Windows 環境建置 4. 實驗設計
簡丞珮	R09922155	1. Feature Selection 文獻回顧 2. Feature Selection 程式實作 3. Mac OS 環境建置 4. 實驗設計