stanCode
標準程式教育機構

Assignment 3

This handout is the product of join work by Jerry Liao and Yi-Ming Tseng
Image Credit: Open Deep Tech

這份作業將派遣各位資料科學家去實戰 Kaggle 競賽題目! 使用 **Classification** 與 **Regression** 預測不同的挑戰。本次作業分成三個部分：第一部分從零開始寫 Python 程式碼、第二部分使用 pandas + scikit-learn 包裹、最後一部分將開放同學互相競賽 Boston Housing Dataset

作業估計時數 - **15** 小時

如果作業卡關歡迎各位到社團提問/運用 TA 時間，也非常鼓勵同學們互相討論作業之概念，但請勿把 code 給任何人看（也不要把程式碼貼在社團裡）分享您的 code 會剝奪其他學生獨立思考的機會，也會讓其他學生的程式碼與您的極度相似，使防抄襲軟體認定有抄襲嫌疑

作業檔案下載

首先讓我們來回顧課堂上提到的鐵達尼號數據集（如表所示）

Survived	該名乘客是否生存
Pclass	乘客所在的艙等(1最好、3 最差)
Name	乘客的姓名
Sex	乘客的性別
Age	乘客的年紀
Sibsp	乘客的兄弟姐妹以及配偶的人數
Parch	父母子女人數
Ticket	票卷號碼
Fare	票價
Cabin	船艙號碼
Embarked	登船的港口

這個數據集其實隱含了相當豐富的資訊，讓同學們從中學習到各種資料處理/分析的技巧！在這份作業中，我們將利用此數據集預測乘客能否在災難中存活? 透過資料重回歷史現場

Level One Abstraction - Classification

第一部份我們將從讀檔、建造資料、甚至建造機器學習演算法 (調參數、微分式、...) 全部自己手刻！由於幾乎所有部分都自己手刻，因此我們稱之為 Level 1 Abstraction (最低階的 Abstraction)。您要編輯的檔案為：**titanic_level1.py**

Milestone 1 - Training data preprocessing

請將 CSV 檔案的資料全部存入一個名為 **data** 的字典中。相信各位同學在觀察數據集時可以發現，**Age** 與 **Embarked** 兩個資訊多有遺漏，因此請同學將資料存進字典時移除有缺漏的資訊（也就是說，請大家移除有一筆 **NaN** 的乘客所有資料！）

請完成 **titanic_level1.py** 檔案裡名為 **def data_preprocess(filename: str, data : dict, mode: str, training_data: dict)** 的函式。**filename** 儲存您要處理的 csv 檔案名稱；請將該檔案的資料整理到 **data** 這個使用者傳入函式的 Python dict。請將 csv 檔案的每個 column 第一列文字當成 **key**（特徵名稱）、**value** 則是該 column 的剩下所有資料（以list的形式保存）。**mode** 是讓 **data_preprocess** 函式知道是 Training Data 還是 Testing Data 的指標文字。若 **mode=='Train'**，請萃取 cvs 檔下列 8 項的資料，並將 **data** return 即可：

Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked

以下三點請同學注意：

1. **training_data** 只有在 **mode=='Test'** 的時候才會用到，請先忽略
2. 請將 Sex 資料底下的 male 設為 1；female 設為 0
3. 請家 Embarked 資料底下的 S 設為 0；C 設為 1；Q 設為 2

完成後，請同學試著跑 **grader.py**！第一項測試是在看「資料個數是否為 712 筆資料」(不是 891! 因為只要 **Age** 與 **Embarked** 有缺的都被丟棄了)。第二項測試是在看「第6筆數據所有的數值、資料型別是否正確」（若錯誤，應該是丟棄 **Age** 資料時出了問題）。第三項測試是在看「第47筆數據的數值、資料型別是否正確」（若錯誤，應該是在丟棄 **Embarked** 資料時出了問題）

Milestone 2 - Testing data preprocessing

在 **titanic_level1.py** 檔案裡名為 **def data_prerpocess(filename: str, data : dict, mode: str, training_data: dict)** 的函式若 **mode == 'Test'**，我們將不會有Survived！感興趣的資料因此剩 7 項：

Pclass	Sex	Age	SibSp	Parch	Fare	Embarked

請讀取以 **filename** 為檔名的 csv 檔，並將上述 7 項資料模仿 **Milestone 1** 的方式整理至 **data** 這個使用者傳入的 Python dict 並在結束後 return data。請注意：**test.csv** 底下**缺失的資料請用使用者傳入的 training_data 這個 Python dict 相對應欄位的資料平均值去填補 (取到小數點下第 3 位)**。也就是說，您在 **Milestone 1** 整理完的 **data** 會在使用者處理 Testing Data 時被傳入 **training_data** 的變數裡

完成後，請同學執行 **grader.py**。第一項測試是在看「資料個數是否為 418 筆資料」。第二項測試是在看「第 11 筆的數據數值、資料型別是否正確」（若錯誤，應該是在填補 **Age** 的資料時出了問題）；第三項測試是在看「第 153 筆的數據數值、資料型別是否正確」（若錯誤，應該是在填補 **Fare** 的資料時出了問題）；第四項測試是在看「 **'PassengerId', 'Name', ...**」是否沒有被加入 **data** 中

Milestone 3 - One-hot Encoding

我們發現 Sex 與 Embarked 是一種類別型的數據。因此我們不能單純只用 0、1 取代 ‘male’, ‘female ’、或是以 0、1、2 取代 ‘S’, ‘C’, ‘Q’ (因為性別、港口並沒有上下關係)。因此我們需要引入 **One-hot Encoding**

One-hot Encoding 意指將類別型的特徵根據其擁有的 N 個類別拆解成 N 個新特徵。以 Sex 為例，**One-hot Encoding** 會將 Sex 變成 Sex_male (以下簡稱 Sex_1) 與 Sex_female (以下簡稱 Sex_0)，並在每個新特徵底下使用 **0** 與 **1** 代表類別的有無。今天如果乘客的性別為 **‘male’**，則乘客在 **Sex_1** 的值為 **1**；**Sex_0** 的值則為 **0**。請同學完成 **def one_hot_encoding(data : dict, feature : str)** 的函式，**data** 代表尚未進行 **One-hot Encoding** 的資料，**feature** 則是需要進行 **One-hot Encoding** 的特徵名稱。舉例來說，**data** 尚未傳入 **one_hot_encoding** 前我們呼叫 **print(data.keys( ))** 再緊接著呼叫 **print(one_hot_encoding(data, ‘Sex’))** 您應該會看到與下圖一模一樣之結果：

```
============Sex One-hot Encoding============
dict_keys(['Survived', 'Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked'])
dict_keys(['Survived', 'Pclass', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked', 'Sex_0', 'Sex_1'])
```

完成後，請同學執行 **grader.py**。第一項測試是在看「Sex 裡原為 **’female’** 的資料是否成功加入名稱為 **‘Sex_0’** 的 one-hot vector」。第二項測試是在看「**Pclass** 裡原為 **1** 的資料是否成功加入名稱為 **‘Pclass_0’** 的 one-hot vector」；第三項測試是在看「**Embarked** 裡原為 **‘S’** 的資料是否成功加入名稱為 **‘Embarked_0’** 的 one-hot vector」；第四項測試是在看「是否有移除原先的 features (如 **Sex**, **Pclass**, **Embarked**)」

Milestone 4 - Normalization

資料前處理的最後一步，就是要將不同的特徵進行常規化！請同學完成 **def normalize(data : dict)** 的函示，將 **data** 中的每個特徵分別進行常規化，使 **data** 的數值都是介於 0~1。請同學執行 **grader.py**，test4_0 測試是在看「Fare 裡的資料常規化後數值總合是否正確」。完成 **Milestone 4** 後，恭喜各位同學完成資料前處理的所有步驟了！

Milestone 5 - Classification Model Training

完成資料前處理後，請先得到訓練模型所需要的 **inputs** 與 **labels**。這部份與 **Assignment 2** 非常的類似！但要特別注意的是 **Assignment 3** 引入了特徵多項式的概念，因此特徵項目會根據 **degree** 的不同而不同。這份作業需要同學實作出 **degree=1** 與 **degree=2** 多項式的特徵，其中 **degree 1** 多項式的特徵同學們在 **Assignment 2** 中就已經完成了！而二次多項式的特徵則是需要同學藉由 **inputs** 間，**各個特徵兩兩相乘而產生**。因此，若 **inputs** 有 **N** 個特徵，則其二次項部份就會有 **(N)*(N+1)/2** 個特徵，整個特徵向量則會有 **(N)*(N+1)/2 + N** 個特徵(包含原本一次項的部分)。請同學完成函式 **def learnPredictor(inputs : dict, labels : list, degree : int,numEpochs: int, alpha: float)**，將 **inputs** 的資料根據使用者指定的 **degree (只會有1與2兩種)** 產生對應的特徵向量，並初始化相應的權重。接著利用 **Gradient Desent** 對權重進行訓練！訓練完成後請回傳 **weights**。另外，歡迎使用我們提供給大家的 **util.py**! 裡面有作業二的正解喔！

完成後，請同學執行 **grader.py**。test5_0 是在看「degree1 的模型是否正確」; test5_1 測試是在看「degree2 的模型是否正確」。如果在不同的 **degree** 下，同學預測的準確率**皆大於80%**，此時的模型就能夠預測船上乘客是否能夠平安存活了!（**Level One Abstraction** 的部分我們不要求同學輸出 Testing Data 的預測值，但歡迎大家試試看有沒有辦法讓上傳 Kaggle 的排名往前進 🏃‍♀️🏃‍♂️）

Level Two Abstraction - Classification

第二部份我們將使用 pandas (簡稱 pd) 讀檔、建造資料再使用 scikit learn (簡稱 sklearn) 做機器學習！由於許多參數與細節都被 pd 與 sklearn 藏起來，因此我們稱之為 **Level Two Abstraction**。您要編輯的檔案為：

titanic_level2.py

Milestone 6 - Training data preprocessing

請將 CSV 檔案的資料全部存入一個名為 **data** 的 **DataFrame** 中（使用 **pd.read_csv( )**）。相信各位同學在觀察數據集時可以發現，Age 與 Embarked 兩個資訊多有遺漏，因此要請同學將資料存進字典時過濾掉有缺漏的資訊（**也就是說，請大家移除有一筆 NaN 的乘客所有資料！**）。

請完成 **titanic_level2.py** 檔案裡名為 **def data_preprocess (filename, mode='Train', training_data=None)** 的函式。如果 **mode=='Train'**，我們感興趣的資料只有下列 8 項：

Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked

請將每個數值正確讀取並**轉成數字**！完成後，請同學執行 **grader.py**：第一項測試是在看 **mode=='Train'** 時「data 是否被整理成下圖的樣子」

Hint: 使用 **dropna( )** 可以直接丟棄一整個欄位

```
Pclass      712
Sex         712
Age         712
SibSp       712
Parch       712
Fare        712
Embarked    712
```

第二項測試是在看 **mode=='Train'** 的時候「**labels** 是否正確無誤」**Survived** 是我們想要預測的目標！因此請同學將 **Survived** 的資料獨立儲存於我們為您創建的 **labels** 變數。

Milestone 7 - Testing data preprocessing

在 **titanic_level2.py** 檔案裡名為 **def data_preprocess(filename, mode='Train', training_data=None)** 的函式若 **mode == 'Test'**，我們將不會有 **Survived**! 感興趣的資料因此剩 7 項：

Pclass	Sex	Age	SibSp	Parch	Fare	Embarked

請注意：**test.csv** 底下缺失的資料請用 **training_data** 這個 **DataFrame** 得到資料的平均值去填滿(取到小數點下第 **3** 位)

完成後，請同學執行 **grader.py**。第一項測試是在看 **mode=='Test'** 的時候「data 是否被整理成下圖的樣子」

```
Pclass      418
Sex         418
Age         418
SibSp       418
Parch       418
Fare        418
Embarked    418
```

第二項測試是在看「第11筆的數據數值、資料型別是否正確」（若錯誤，應該是在填補 **Age** 的資料時出現了一點問題）；第三項測試是在看「第 153 筆的數據數值、資料型別是否正確」
（若錯誤，應該是在填補 **Fare** 的資料時出現了一點問題）

Milestone 8 - One-hot Encoding

請將 **'Sex', 'Pclass', 'Embarked'** 使用 one-hot encoding! 也就是說，請加入 **'Sex_0', 'Sex_1', 'Pclass_0', 'Pclass_1', 'Pclass_2', 'Embarked_0', 'Embarked_1', 'Embarked_2'** 這幾個由 one-hot encoding 萃取出來的新項目

請同學執行 **grader.py**。第一項測試是在看「**Sex** 裡原為 **female** 的資料是否成功加入為 **Sex_0**」。第二項測試是在看「**Pclass** 裡原為 **1** 的資料是否成功加入為 **Pclass_0**」; 第三項測試是在看「**Pclass** 裡原為 **2** 的資料是否成功加入為 **Pclass_1**」; 第四項是在看「是否有移除原先的 features (如 **Sex**, **Pclass**, ...)」

Milestone 9 - Standardization

除了 Normalization，資料前處理中更強大的工具莫過於 Standardization！請同學完成 **def standardization(data, mode='Train')** 這個函數。將 **data** 中的每個特徵分別進行標準化。為了將資料標準化，我們可以輕鬆呼叫 sklearn 包裹裡面的 **preprocessing.StandardScaler( )** 來幫我們處理！請同學執行 **grader.py**，test9_0 測試是在看「**Fare** 裡的資料標準化的數值總合是否正確」。
(請注意：**Milestone 9** 只是要測試同學們的觀念是否正確，您將不需要在 **Milestone 10** 呼叫 **def standardization(data, mode='Train')**)

完成**Milestone 9** 後，恭喜各位同學完成資料預處理所需要的所有步驟了！

Milestone 10 - Classification Model Training

最後我們將使用 **sklearn** 來完成最後 Training & Testing 的部分！請大家使用 **Milestone 6** 寫好的 functions 合併到 **def main( )** 裡，讓 **linear_model.LogisticRegression(max_iter=10000)** 可以成功跑出結果！

大致的流程如下：

(a.) 呼叫 **data_preprocess(TRAIN_FILE)** 來處理 NaN

(b.) 對 **'Sex', 'Pclass', 'Embarked'** 分別呼叫一次 **one_hot_encoding( )**

(c.) 直接將所有資料標準化 (不要呼叫 **Milestone 9** 的 **standardization( )**)

(d.) 呼叫 **preprocessing.PolynomialFeatures** 來達到高維度的特徵！

(e.) 呼叫 **linear_model.LogisticRegression(max_iter=10000)** 並開始 training

(f.) 跑出準確率的確切數值

degree 1 的 Training Accuracy 大約為 ~80% ; 而 degree 2 的 Training Accuracy 會是 ~83% ; 而 degree 3 的 Training Accuracy 會是 ~87%! 請同學將確切的小數點下 **8** 位的數值填寫在 **def main( )** 註解區

```
def main():
    """
    You should call data_preprocess(), one_hot_encoding(), and
    standardization() on your training data. You should see ~80% accuracy
    on degree1; ~83% on degree2; ~87% on degree3.
    Please write down the accuracy for degree1, 2, and 3 respectively below
    (rounding accuracies to 8 decimals)
    TODO: real accuracy on degree1 -> _______________________
    TODO: real accuracy on degree2 -> _______________________
    TODO: real accuracy on degree3 -> _______________________
    """
    pass
```

Open Ended Challenge - Regression Problem

最後一個部份我們將指派各位厲害的資料科學家們去波士頓預測房價！作業三內有一個名為「**boston_housing**」的資料夾裡面的 **train.csv** 提供給同學們建造自己的機器學習 Model。

最後同學們要預測的資料為 **test.csv**。請同學們將分析 test.cvs 的結果輸出(格式請參考submission_example.csv)並將答案上傳至 kaggle 競賽連結 https://www.kaggle.com/c/sc201-nov-2021/。

Milestone 11 - Boston Housing

同學們要分析的資料解釋如下

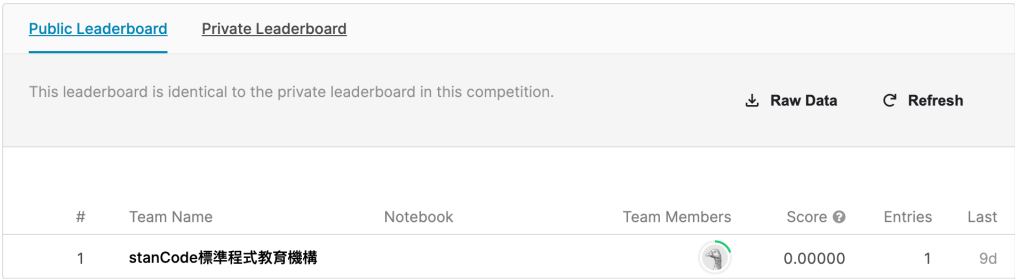
crim	城鎮人均犯罪率
zn	住宅用地超過 25000 sq.ft. 的比例
indus	城鎮非零售商用土地的比例
chas	查理斯河空變量（如果邊界是河流，則爲1；否則爲0）
nox	一氧化氮濃度
rm	住宅平均房間數
age	1940 年之前建成的自用房屋比例
dis	到波士頓五個中心區域的加權距離
rad	輻射性公路的接近指數
tax	每 10000 美元的全值財產稅率
ptratio	城鎮師生比例
black	1000（Bk-0.63）^ 2，其中 Bk 指代城鎮中黑人的比例
lstat	人口中地位低下者的比例
medv	自住房的平均房價，以千美元計

這個部分的自由度非常高！歡迎同學們不管使用自己手刻的模型、pandas 或是利用 sklearn **預測 medv** 都可以！以下四點請同學們注意：

1. 請同學編輯 **boston_housing_competition.py** 檔案，自行讀檔、分析、輸出與 **submission_example.csv** 格式一樣的檔案上傳 kaggle

2. 同學在繳交作業時請確保 **boston_housing_competition.py** 檔案裡面的 code 就是最終上傳 **Public Leaderboard** 的版本

3. 同學每天上傳次數上限為 20 次，由於是競賽，Kaggle 不會將排名顯示（如下圖所示，上傳的答案 score 永遠是 0.000000）



由於是 Regression Problem (**請勿使用分類的 LogisticRegression**)，這邊 score 的計算採取的方式為 RMS（把 L2 Loss 的總和相加再開根號）想知道關於 RMS 更詳細的內容歡迎參考下方連結：http://www.mathwords.com/r/root_mean_square.htm#:~:text=To%20find%20the%20root%20mean,is%20the%20root%20mean%20square.

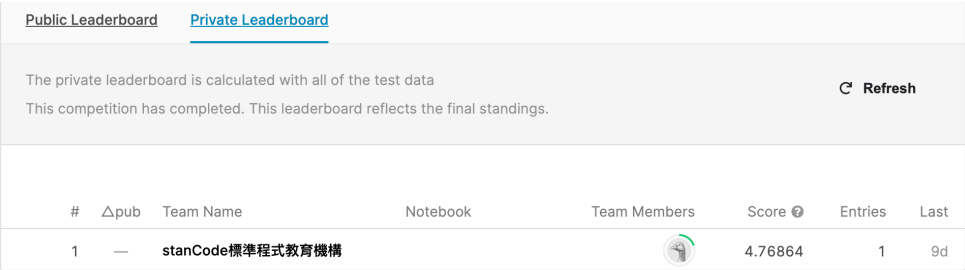
若同學們想要用 sklearn 計算 RMS Error，可以先呼叫 **from sklearn import metrics** 再使用 **metrics** 底下的 **mean_squared_error** 將您輸出的預測值 (**predictions**) 與 True Labels (**y**) 傳入。（如下圖的程式碼所示：

```
from sklearn import metrics
print(metrics.mean_squared_error(predictions, y)**0.5)
```

4. 在 **Testing Set** 數據答案是未知的情況下，同學們一定要自己製造 **Validation Set** 來測試模型是否 Overfitting! 製造 **Validation Set** 的程式碼如下 (若同學想拿 50% 的 **Training Set** 來當 **Validation Set**)：

```
from sklearn import model_selection
train_data, val_data = model_selection.train_test_split(data, test_size=0.5)
```

5. 然而，主辦方可以在 **Private Leaderboard** 看到大家的表現 👀（如下圖所示，其實 stanCode 上傳答案的 score 為 4.76864）



因此我們會在社團更新最新的 **Private Leaderboard** 排名！歡迎大家努力一直往前進 🏃‍♀️🏃‍♂️ 我們會在 **1/23 (日) 23:59** 關閉競賽，並結算最終贏家頒發 500 獎學金給冠亞軍 🏆

🎉 **Congrats on finishing SC201 Assignment 3** 🎉

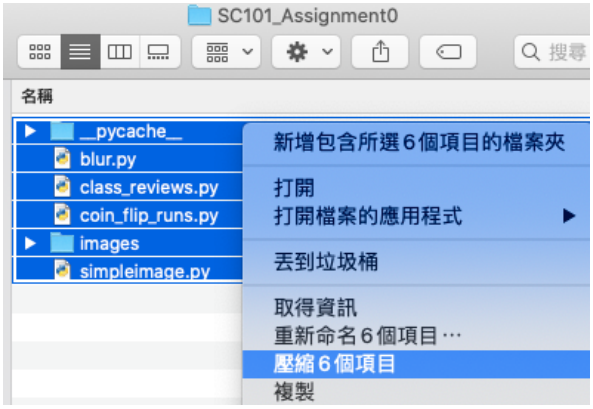
作業繳交

恭喜各位同學完成 SC201 Assignment 3! 這份作業將帶領同學前往更艱深的觀念& AI 世界。這份作業非常實作導向，希望讓完成的同學外來只要來到資料都能做出厲害的模型，成為正港的資料科學家 👩‍🔬 👨‍🔬

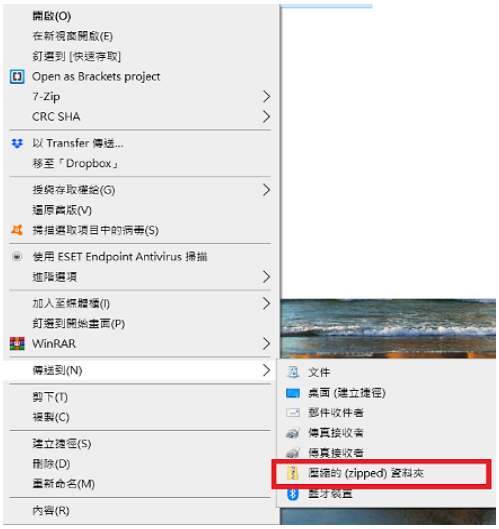
請同學依循下圖，完成作業上傳。

1. 以滑鼠「全選」作業資料夾內的所有檔案，並壓縮檔案。請見下圖說明。

macOS：按右鍵選擇「壓縮n個項目」



Windows：按右鍵選擇「傳送到」→「壓縮的(zipped)資料夾」



2. 將壓縮檔(.zip)重新命名為「a(n)_中文姓名」。如：
   Assignment 1命名為Assignment1_中文姓名;
   Assignment 2命名為Assignment2_中文姓名; …

3. 將命名好的壓縮檔(.zip)上傳至Google Drive（或任何雲端空間）

   1) 搜尋「google drive」
   2) 登入後，點選左上角「新增」→ 「檔案上傳」→ 選擇作業壓縮檔(.zip)

4. 開啟連結共用設定，並複製下載連結

   1) 對檔案按右鍵，點選「共用」
   2) 點擊「變更任何知道這個連結的使用者權限」後，權限會變為「可檢視」
   3) 點選「複製連結」

5. 待加入課程臉書社團後，將連結上傳至作業貼文提供的「作業提交表單」