1. **Classifier Used**

這次的實驗作業我建立了兩種supervised模型去預測我的資料，其一就是Logistic Regression，另外一個就是K-Nearest-Neighbors (KNN)，以下簡述一下兩者的運作原理：

1. **Logistic Regression**

Logistic Regression 是利用Linear Regression的方式先將資料點做運算再代入Logistic Function，像是sigmoid function然後將資料根據機率做預測分類。而訓練的過程就是利用gradient descent的方式去調整Linear Regression的參數，最後在預測階段就將訓練好的參數代入式子中對資料點進行預測。

1. **K-Nearest-Neighbors (KNN)**  
   K-Nearest-Neighbors這個模型特別的一點就是他沒有訓練過程，它對資料進行預測純粹就以樣本空間(以這次作業而言為X\_train)中和受測點距離最近的k個鄰居，針對這k個鄰居的標籤取眾數去決定它的預測結果，而計算點與點之間距離的方法有很多種，而我利用的是最典型的方式：Euclidean Distance，也就是平方合開根號距離。
2. **Data Preprocessing**

在討論data pre-processing以前，先在此簡單描述一下資料的存取方式，根據這次作業目標，我們要針對12個特徵(一些骨頭角度等資料)去預測有沒有背痛(0或1的二分類)，X\_train、X\_test為二為矩陣，每個row存一筆資料，而每個column為一個feature(一共12個欄位)，y\_train則為一個一為陣列，每個位置對應X\_train的資料列位置，存取0或1。

針對這次的資料，我有嘗試進行了以下步驟：

1. **Shuffle**

因為因應後面會提到的cross validation，為避免raw data有分布不均的可能性，我在讀入資料後有將一筆一筆的資料重新洗牌增加隨機分佈的特性，以免影響資料的訓練和模型的優化。

1. **Normalize**

根據每一行(每一個特徵)，將資料點減去最小值除以極值差(最大值減去最小值)，如下公式：

這樣可以將資料分佈於0和1之間，這樣是避免不同特徵在數值上面有不同的區間範圍(例如正負號之差)差異影響訓練的參數和結果。

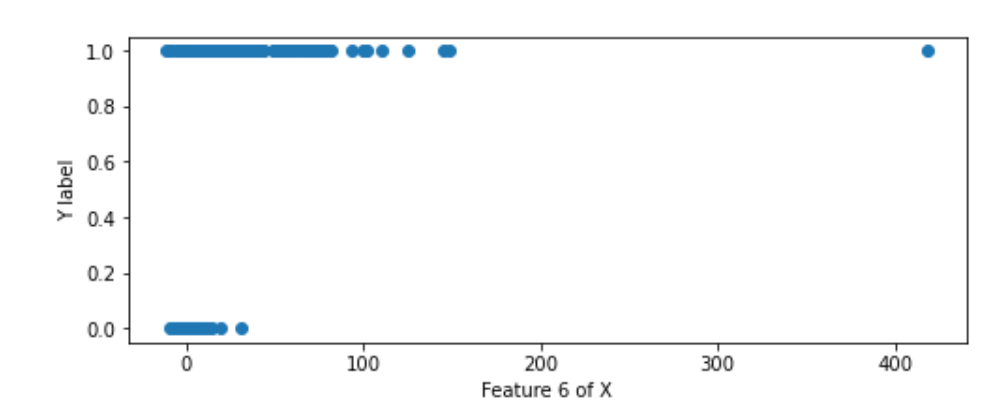
1. **Standardize**

根據每一行(每一個特徵)，將資料點減去行平均值除以行標準差，如以下公式：

將資料標準化後，每一個欄位的資料平均會是0，標準差呈1，之所以會對資料進行標準化是要避免不同的特徵單位(例如數值大小呈倍數差異)影響訓練的參數和結果。

1. **Remove Outliers**

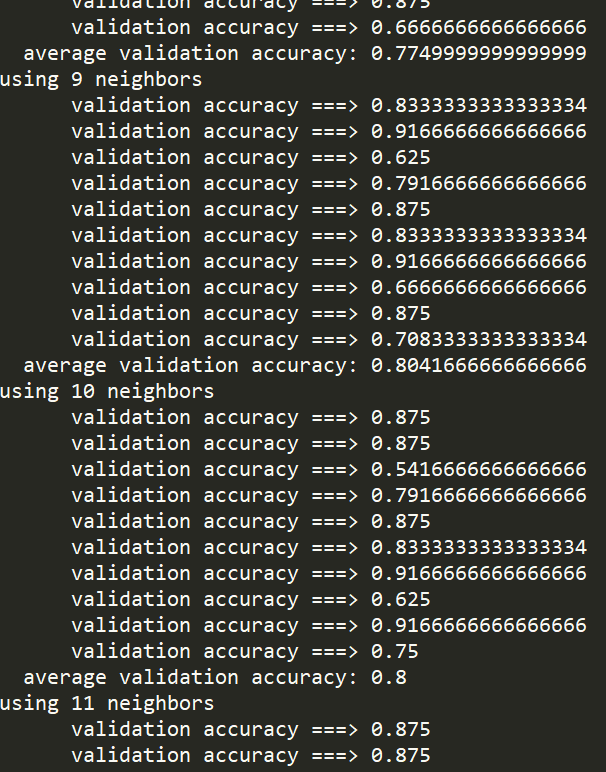
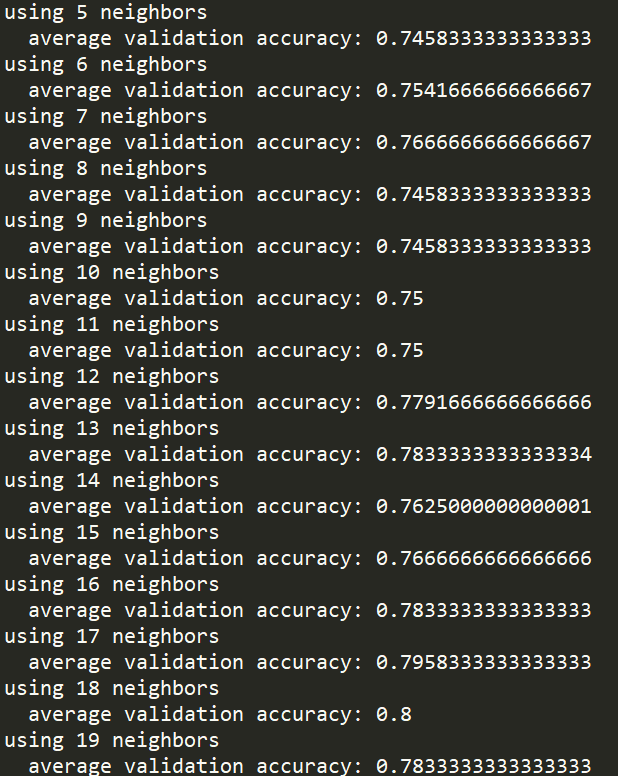
在進行訓練以前的另一關鍵要點就是替除資料中的離群值，避免離群資料(像是錯誤資料、奇異資料點)對訓練有影響造成偏差，因為這次的資料有13個維度有點難視覺化，所以我就產出二維圖形一一觀測單一特徵中資料和分類的分佈關係，舉下圖為例：



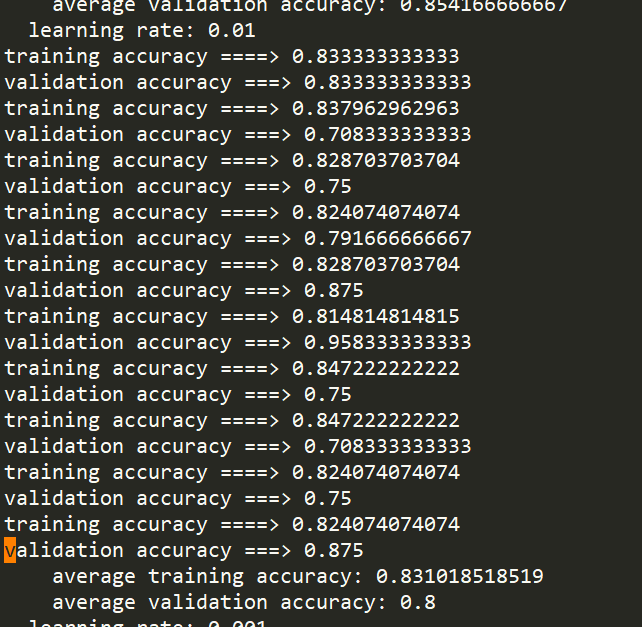
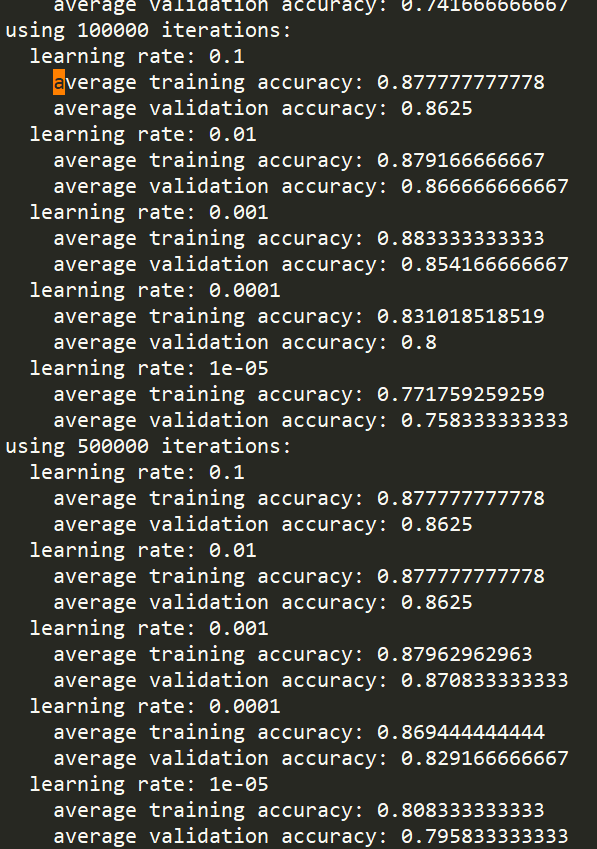
離群

可以從第六特徵和分類關係圖中看到右上角有一數據點和其他資料點相距比較遠，這樣可以看出x中的這一點以第六特徵而言有離群的可能性。為了進一步分辨離群值，我建了一個z score的矩陣，把threshold設為3，Z值大於3的資料都會被塞選出來，再把這些選出來的資料剃除就將資料集中的離群值在初步的處理階段移除了。

1. **Improve Performance**
2. **Train-test split**  
   一開始我將X\_train資料以3:1的比例再切成訓練和測試資料條參數，但後來發決這樣的模式太慢了，而且一旦測試和訓練資料有不均的分配就很容易照呈誤差，所以改用了接下來會提到的cross validation。
3. **Cross-validation**  
   這個方法適合用在資料量少的資料集上(像我們這次的訓練資料就只有兩百多筆)，原則上就是將資料切成n等份，輪流拿一等份當測試資料集 (validation set)，剩下的n-1等份當訓練資料集。我這次就是利用這個方法進行模型的比較以及模型參數的優化。(本次作業我將n設為10)
4. **Cross Validation on K-Nearest-Neighbors**  
   下圖為調k參數的優化過程，適用取不同鄰居數看會不會稱加預測命中機率，過程可以看到跑十次分別的測試命中率以及最終的平均命中率，以右圖得比較可看出，整體平均數大約介於75~80%之間，而取k=16、17會得到較高的預測命中率。

1. **Cross Validation on Logistic Regression**  
   我對於Logistic Regression相較KNN調的參數比較多，一樣的，我先對learning rate還有training iterations進行優化，普遍上來說預測命中率可以落在80~88%之間，略高於KNN的cross validation。

1. **Finalize and Choose Model**  
   KNN : 上Kaggle跑得到最好的結果為 0.72093%  
   Logistic Regression : 上Kaggle跑得到最好的結果為 0.83720%  
     
   從上面做cross validation比較以及Kaggle的結果來看，Logistic Regression具有比較高的命中機率，原因或許是因為KNN需要龐大的資料集材可以做較準確的預測，而我們這次的資料集資料量較少，所以或許這是為甚麼LR會有較好結果的原因。
2. **Difficulties and How I Overcame Them**
3. **Building My Own Models from Scratch**  
   本次作業遇到的困難首先當然就是要先了解模型，因為有規定不能呼叫scikit learn所以很多模型都必須手刻，但好家在numpy的函示庫夠給力，在建立模型和做運算的過程中給了不少幫助，網路上也有不少機器學習文章有講解模型背後運作原理，讓我可以比較快上手。

1. **Inconsistent Data Pre-processing**  
   很多資料處理的細節上一開始都會忽略，就導致自己的預測模型很多bug，像是training data有做standardization而testing沒有，就讓training的accuracy和Kaggle上的差10~20%，一開始以為是overfit的原因，但做完cross validation後才發現原來是自己資料處理前後不一貫，疏忽所導致，也讓我理解到做是最好齊一，先將data都一併做處理再進行訓練才不會落掉。
2. **Python Versions**

Python版本也是個要注意的點，姑且不論python 2和python 3的差別，很多3.\*的版本就有很多細節不大一樣，像是statistics.mode()這個function，它的功能是選出矩陣中出現次數最頻繁的值，而當有兩個以上一樣最多的值在python3.8以前的版本會丟出一個error，但3.8之後就不會，像這樣細節上的差異容易造成在不同平台上跑出現bug (自己電腦v.s server)，所以我發覺要多利用try、except去解決類似的問題。