Pre-processing:

在建立模型以前，對資料做預先處理是非常重要的。原始數據中的異常值、數值範圍、資料缺失等因素可能會很大程度地影響分類器的模型表現。讓我們來一個一個討論是否需要執行以上三種預先處理技術(以下統計數據皆經由Excel計算得出)：

1. outlier detection：在常態分佈下，異常值檢測的一個常用方法是檢測該數據集中是否有數據位於平均數三個標準差開外。首先，我們先計算所有屬性的標準差(在本proposal中，均以前10個屬性為例，若需所有原始數據的統計資料，可見附件)：



第二，我們可以先行簡單地用MIN、MAX、AVERAGE函數來獲取每個屬性的最大值、最小值以及平均數：





將以上四行數據進行比較，毫無疑問地，每一筆資料都存在著異常值，這明顯可能會影響到分類器的表現。不過直接用肉眼觀察可能不夠嚴謹，因此再次使用Excel中的函式：

=SUMPRODUCT((A2:A1601 > AVERAGE(A2:A1601) + 3 \* STDEV.P(A2:A1601)) + (A2:A1601 < AVERAGE(A2:A1601) - 3 \* STDEV.P(A2:A1601)))

這個函式將由Excel計算。將計算每個屬性中總共有幾個異常值存在。以下是前十筆資料的計算結果：



事實上不僅僅是前十個屬性，可以說所有屬性都有異常值的發生。因此，儘管我們仍需透過交叉驗證來決定是否為原始資料進行異常值檢測。但我認為，對原始資料進行異常值檢測是相當有必要的。

至於對異常值的處置方式，一般來說通常有兩種：刪除該筆資料，或是將異常值改為平均數。在決定處理方法之前，我們可以先計算出每個屬性的異常值發生機率：



以及其最大與最小發生機率：7.25% / 4.375%。

一般來說，異常值為7.25%~4.375%的比例屬於可接受範圍內，是可以刪除異常值發生的該筆資料的。儘管如此，為了更好的分類器，還是可以對其進行交叉驗證來選出對異常值最佳的處置方式。

1. Normalisation：資料的歸一化用於避免某個屬性其權重遠大於其他屬性的情況。通常發生於某屬性的domain特別大的時候。我們同樣可以先利用Excel來獲取每個屬性的數值domain(由屬性最大值 – 屬性最小值產生)：



計算完成後擷取出此1600個屬性的最大與最小domain：1371.19、0.095。

經由以上，我們可以得出資料的標準化也是相當重要的，否則屬性值domain越大的(例如1371.19)可能會主宰分類器邏輯，而越小的(例如0.095)可能會被分類器邏輯忽視，但這並不代表他們不重要或是特別重要。每個屬性應當要有公平的權重。

1. Imputation：Imputation的處置較為simple。對於所有有缺失的資料，我們有幾種處置方式：直接忽視、插入所有資料的平均值、插入與該筆資料同一個class的該屬性平均值。直接忽視的方法顯然我們也可以”直接忽視”，但對於剩下兩種方法，乍看之下也許插入同一class的該屬性平均值為最優解。但事實上我們仍需對此兩種方法進行交叉驗證用以確定哪一種處置方法可以使得分類器運作地更好。

Application of Classification Techniques：

從第二週至第八週的教學內容我們可以得出decision tree、random forest、k-nearest neighbour and naïve bayes這四種方法是此case之下的潛在分類器邏輯。沒有任何一種方法是壞方法，重要的是我們要透過多次交叉驗證來確定使用什麼方法以及其hyperparameter。

* Decision tree：利用屬性將資料做分組，無論是root node或是non-root node都有屬於自己的停止標準，如果沒有達到停止分類標準就將繼續分類下去。Decision tree的運作邏輯相當好理解，其難點為要用哪個屬性來作為下一個分類標準，在這之中就會牽涉到交叉驗證。交叉驗證主要用於選出利用哪個purity measures來做為構建Decision tree的方法。

Decision tree是一個經由二分類或是多分類法來對資料進行預測分類的方法，在原始資料尺寸較小時能夠有較好的預測表現。對於較大的數據集或是較大的數據維度可能會造成Decision tree有很大的深度，訓練時間可能會大幅增加。

* Random forest：Random forest雖然與Decision tree都是以”tree”的樣子來進行分類，但是兩者的運作邏輯大有不同。Random forest側重於”隨機的概念”，包括從

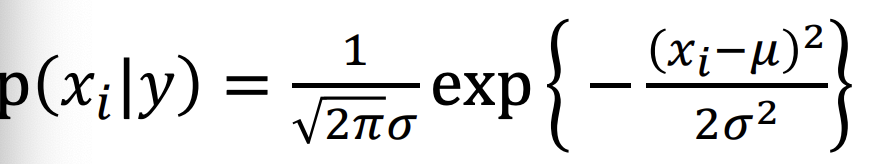
training data中隨機抽取樣本以及屬性來訓練模型，而非暴力地使用所有原始數據。也正因為其隨機且多量(森林)的特性，Random forest對於噪點以及異常值擁有較高的耐受度。

另外，歸功於其”集合”的架構，Random forest比起Decision tree能夠處理較大尺寸以及維度的資料，因為有多個Decision tree能夠分散需要處理資料的數量。

* 𝑘-Nearest Neighbours：在此方法中，有兩個特別重要的因素需要考慮。也就是該使用何種距離衡量方法來評斷點與點之間的距離。根據此case的描述，若有某一個點她的第106個屬性為1，也就是這個基因具有細胞通訊功能。那麼越靠近此點的其他資料點理應來說在分類上應與此點越接近。即若兩點距離相當近僅有些許數值差別，應該將其分類為同一個類別而非不同類別。根據以上敘述，我們在此專案中可能採取曼哈頓距離。至於k值的決定，除了遵守老師課堂中提到的”k數值一般情況下需要為奇數”外。我們還需要透過交叉驗證來決定哪一個奇數。

儘管𝑘-Nearest Neighbours擁有許多優點，方法簡單且直觀，也就是計算距離。且對於噪點以及異常值有一定的耐受度。但是，由於Curse of Dimensionality，𝑘-Nearest Neighbours在面對相當高維的數據時表現會相當不佳。

* Naïve Bayes：對於全是連續小數的這個case來說，使用Naïve Bayes可能會有可觀的計算量。但是邏輯也相對單純。我們只需要透過老師在上課提到的公式：



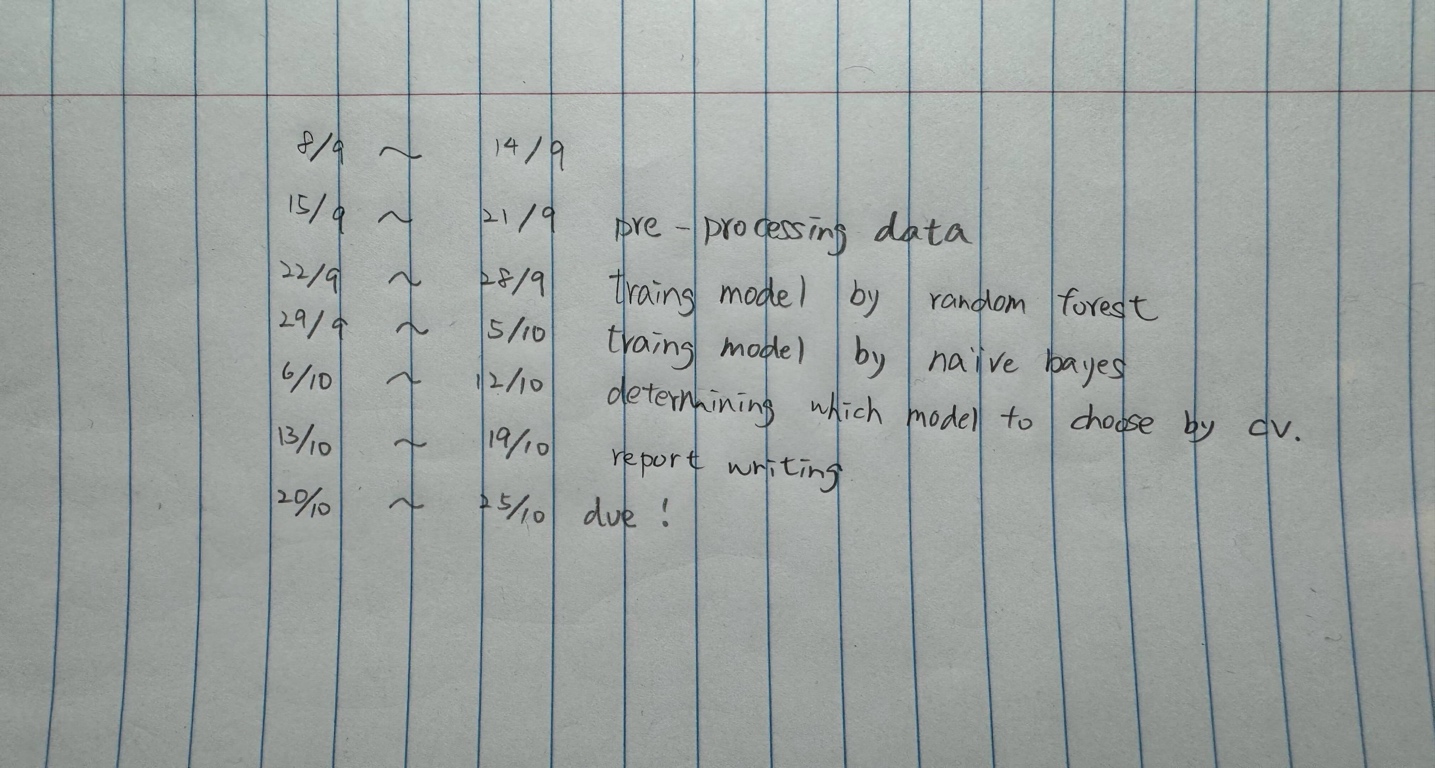
任何新資料進入後，令y為1(第一個class)。遍歷所有屬性x。xi為當前屬性的，將後面的公式更改為xi屬性的對應統計數值，計算出p(xi | y), i = 1, 2, 3, 4, …, j各為多少之後相乘，得到的數字與令y = 0(第二個class)且後續計算方法相同得出的數字後做比較。較高的即為模型預測答案。顧名思義，Naïve Bayes是屬於比較簡單單純的計算方式，因此在設計模型內部時並不特別考慮hyperparameter。

Naïve Bayes非常適合用於擁有高維度的數據集，且算法邏輯較簡單。但要注意的是，Naïve Bayes假設各個屬性都具有獨立性，在這個case中，我們因並不知道105個屬性中的依賴關係，因此在使用時必須要較為留意。

Model Evaluation：

有鑒於Application of Classification Techniques所提到的概念，就我個人觀點認為，在這個case之下採用Random forest或是Naïve Bayes可能是可行的策略。他們在處理較為大量且高維度的原始數據量的時候都能保持相對良好的模型表現。採用投票表決(Random forest)以及比較class的機率(Naïve Bayes)的運作方式也較為有彈性而不一口咬定預測結果。值得提及的是，在此case之下Naïve Bayes的建構邏輯較Random forest單純許多，所需成本與時間也較低，但要注意屬性之間必須要有獨立性確保Naïve Bayes能正確執行。在最為嚴謹的情況之下，若時間允許，對此兩種方法進行交叉驗證來決定最終選定方法是理想的。上面提及我們將採用82原則(也就是80%的資料作為training data, 20%的資料作為testing data)。交叉驗證的過程為，對於每一個classification techniques，將第一份資料作為testing data，其餘四份為training data。接下來是第二份資料為testing data，其餘四份為training data依此類推。在每一次的迭代中，訓練完成後利用testing data以獲取其性能指標(accuracy, z-score)等等。最後將所有性能指標做統計計算以獲取其平均性能指標以及標準差。計算出所有classification techniques的平均性能以及標準差之後，將其相互比較，儘量獲取越大的平均準確度以及越小的標準差來選取模型表現最好的方法來最終classification techniques。

Project Timeline：



基本上，對於此專案我可能會利用python作為建構程式語言。雖然本人對於Java可能較為熟悉，但Java這種物件導向的程式語言可能不適合用在這個模型上。Python一體成型、一鏡到底的運作模式可能會比較適合此專案(線性專案)。以上的圖片就是timeline。首先，對資料進行預先處理是非常必要的，這能確保資料的品質並提高模型表現性能。其次，分別利用random forest以及naïve bayes來訓練模型。第三，利用交叉驗證來選出哪一個方法能有更好的模型表現。並在花費一個禮拜的時間完成對此模型的report。