大家好，我們是教父，我們這次報告的主題是:見微知著:讓python成為你的股票理專，以下是我們的組員名單，我是資訊系112的莊上緣，接下來由我先來為各位報告。

首先，我要來為各位簡短地介紹這次我們的final\_project大致上要做甚麼。我們希望利用幾檔較具代表性的股票來做出股價預測模型，並搭配UI介面設計成一個可以與使用者互動的程式。像這張投影片的背景圖就是我們這個程式的UI主畫面，使用者可以輸入股票代碼，想拿來預測的日期，就可以印出股價折線圖，並且左上角還可以依照使用者偏好來做圖表外觀上的調整。接下來我們請另一位組員來為各位介紹後續的部分。

好的，接下來也是由我來為各位介紹我們這次主要選擇的機器學習演算法。由於股票的許多數值波動都是跟時間相關的，因此股票是一種時間序列資料。所以我們選擇了LSTM這個可以支援時間序列模型相關預測的機器學習演算法來幫助我們。我們會透過import keras這個package來幫助我們搭建一個LSTM神經網路，keras是一個專門用來訓練深度學習模型的API，在這次專案裡面我們使用的是sequential模型，從字面上看來，這似乎是一種簡單的線性序列模型，其實不然，他應該被理解為一種透過堆疊很多隱藏層來建構出的深度神經網路，在神經網路中，每個神經元皆會配有一個權重，這個權重代表著該神經元之重要性，而這個權重會在神經網路模型每次學習時，根據學習結果與大量輸入的資料而進行不斷地更新，我們必須建立輸入輸出控制與隱藏層，選擇適當的優化器與損失函數。而程式碼中的input\_shape其實是一個三維資料，代表batch size,time step,input data, batch代表一次要輸入的資料筆數，time step 代表資料的時間維度，股票如果要藉由前10天的資料預測，則time step為10。Input data代表單獨一個時間的資料，而units為每一個step輸出的數量，當要搭架多層的LSTM時，return sequences設置為TRUE，這樣的步驟又稱為sequence-to-sequence。接著，我們挑選ReLU(Rectified Linear Unit)作為激勵函數，它有兩個優點，一，可以幫助我們解決反向傳播訓練時產生的梯度消失問題，二，使部分神經元輸出為0，讓神經網路變得稀疏，我們也一邊使用dropout隨機消除神經元，緩解過度擬和的問題。

損失函數(loss function)可以幫助我們做現在ML模型好壞的評估

並以數值呈現。

當資料完成了一次完整的訓練，這個過程稱為一個epoch。我這邊為各位舉個例子，方便說明batch size,number of iteration與Epoch的關係

比如今天有400筆資料，我一個梯次想丟40筆資料進去，那我的batch size就是40，而迭代次數則是10，也就是資料總數400除以batch size40`,那經過了10次迭代後，代表我們掃過了所有資料一遍，這稱為完成了一個epoch

Epoch的數量增加

加，會使神經網路中權重的不斷更新，一般來說，當訓練次數增加，準確度會較高。

我們在這次專案中有使用到stack LSTM，相對於單層的LSTM來說，這個堆疊LSTM的每個神經元具有更多的層數。為了將一個深度學習模型優化，原理是減少總LSTM神經元數，相較於較多的神經元與深度較淺的神經網路模型，深度較深，較多層的神經元組成的模型，更能近似地代表更多數的函數，也就是說，一個深層且分層的神經網路模型會比一個淺層且大的神經網路模型更能代表某些功能。

也就是說，神經網路的深度使其在具有廣泛挑戰性的預測問題中取得成功。

接著我要為各位介紹的是我們如何切分資料的，我們以長榮公司這檔股票做為我們的dataset來源，我們一開始的raw data的time interval是取5年，從2016的6月4日取到今年的6月4號，因此我們決定嘗試看看使用4:1的方式去分配資料中用來train的比例與用來驗證的比例。

接著我們來看看方才介紹LSTM的時候提過的dropout size參數，我們可以搭配左下角這張圖來看，原本的一層五個的神經元與其他神經元皆有錯縱複雜的關係，這可能導致該模型不同神經元之間的相互影響變得過於劇烈導致模型測試效果不佳，因此我們將其中幾個神經元drop掉，可以看出，在每層drop掉百分之四十的神經元數量，他們之間複雜的關係問題得到了大大的改善，神經元之間的連線變得較為單純。我們這邊也有做試驗，發現當dropout size為0.4時，train出來的效果是最好的。這邊牽涉到三個迴歸評價指標，分別是RMSE、EVS與r-squared，首先，RMSE是root mean squared error，也就是均方根誤差，其實就是將大家都學過的MSE開根號，而EVS就是explained variation score 可解釋變異，代表給定的數據中的變異能被數學模型所解釋的部分，通常會用方差來量化變異，故又稱為可解釋方差。最後，r-squared稱為判定係數，代表可解釋變異佔總變異的比率。待會後面的其他參數測試也會以dropout size等於0.4的狀況下去測試。

再來，我要為各位介紹的是我們透過上面那些步驟預測出的模型之各項精準度評比。為了方便製作confusion\_matrix，我們決定將股票的漲跌趨勢做數字代號分類，我寫了一個副程式，用來計算時間段中的真實股價漲跌趨勢與預測股價漲跌趨勢，比較每一天與前一天的股價差別，漲價為1，下跌為0，而若維持不變則為2，左下角是我們的真實股價與預測股價的摺線圖，黑線為真實股價，綠線為預測股價，我們將剛剛副程式算出的真實與預測股價之趨勢list匯入sklearn提供之confusion\_matrix並得出如右下角的結果。可以看出這個模型的accuracy為接近6成。接下來由另一位組員為各位介紹其他的evaluation and result。

好的，接下來由我來為各位報告剩下的部分，我們當初會選擇以十天做為一個step size，每十天就為其放上一個標籤，是因為以我們目前對於股票的理解，若只有三五天好像太短，但是若將時間拉到一次兩周或是一個月則似乎又太長，因此我們取10天，但是後來我們發現這個step size似乎也跟每檔股票的個性有不小的關係，因此我們可以再多做嘗試。接著是我們推測，為何我們使用了多變量時間序列模型，但是accuracy卻還是不慎理想的一個可能的原因，我們認為可能發生了多重共線性，因為在現實生活中，很難找到一組互不相關，又對因變數y產生重要影響的變數，我們在訓練時丟入的許多特徵值可能也是因彼此不完全是不相關，而產生了多重共線性的問題，我們在網路上找到了一個看起來可行的解決方案，就是drop掉其中幾個變量，我們發現他的原理，有可能跟剛剛的dropout很類似，都是將一群互相之間關係錯縱複雜的預測因子drop掉幾個，讓其之間的相互影響得到有效的減低。

報告的最後，這邊就是我們這組的分工與貢獻表，接著，我想請我們的一位組員來稍微demo一下我們的UI程式

(DEMO)

我們的報告就到這邊，感謝各位的聆聽。