Final Project Report

1. Members

- 統計 112 劉恩兆 B14081027
- 統計 112 宋穎恩 B14081302

2. Project Title

- What’s Cooking? Use recipe ingredients to categorize the cuisine
- Competition on Kaggle：https://www.kaggle.com/competitions/whats-cooking/overview

3. Problem Statement

在此競賽中，我們需要利用食譜中的食材組成，預測相對應的料理種類。

因此，此資料集的 Input 為各種不同的食材組成，而 output 則是不同國家的料理種類。

表一、資料集 input & output 示意圖

X (input)：ingredients	Y (output)：cuisine
• Sugar • Hot chili • Asian fish sauce • Lime juice	• Thai

而此競賽是以 Accuracy 來進行評分，公式如下：

分數計算方式為：$\frac{\text{正確分類筆數}}{\text{總筆數}}$

4. Solved Challenges

以下三點為我們預期會遭遇到的技術挑戰，最終皆有找到相應的解決方法。

4-1. 資料維度的疑慮

經計算，不重複的食材有 6,714 種，若是直接用 one-hot encoding 可能會使資料維度過大而造成硬體不足，又或是使資料過於分散，導致模型效果不盡理想。

4-2. 資料處理

處理 input 資料有很多種方法（如：詞向量、 one-hot encoding、label encoding 等），需多加試驗已找出最合適的處理方法。

4-3. 模型選擇

此資料集有許多模型種類可以使用，如何挑選適合的模型種類，以及如何調整模型以提高準確率都需要多加探索。

5. Used Dataset

主要使用的資料集為 Kaggle 提供的 train.json 以及 test.json，其中：訓練集有 39,788 筆資料，三個欄位；測試集有 9,944 筆資料，兩個欄位，而相差的欄位即為訓練集中的「cuisine」欄位，也就是需要在測試集預測的欄位。

表 2、訓練集示意圖

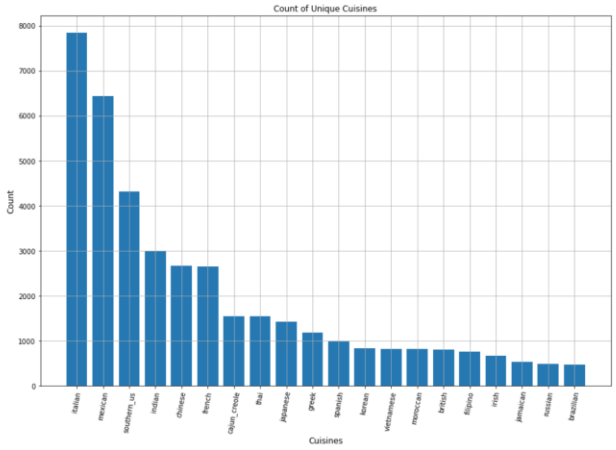
id	ingredients	cuisine
10259	[romaine lettuce, black olives, grape tomatoes, garlic, pepper, purple onion, seasoning…]	Greek

表 3、測試集示意圖

id	ingredients	cuisine
18009	[baking powder, eggs, all-purpose flour, raisins, milk, white sugar]	To be predicted

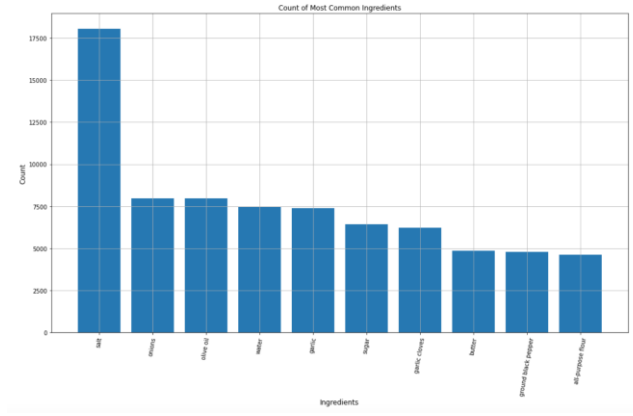
6. Explorative Data Analysis

圖 1 、Cuisines 種類統計



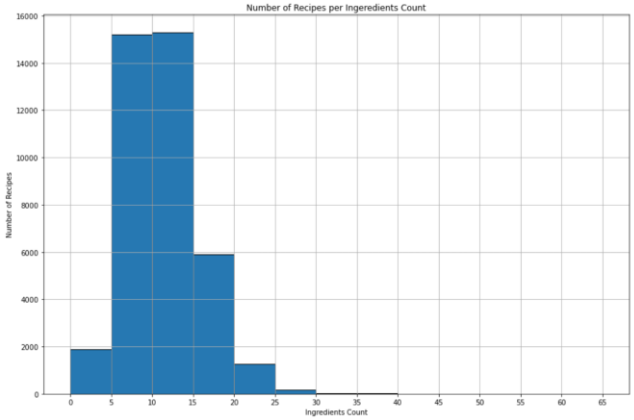
由圖 1 可以得知，此資料總共有 20 種不同的 Cuisines，其中出現次數由多至少的前三名依序為 Italian (義大利菜)、 Mexican (墨西哥菜) 與 Southern_us (南美洲菜)。

圖 2、Ingredients 種類統計



由圖 2 可以發現， Ingredients 出現次數由多至少的種類前三名依序為 Salt、Onions 與 Olive oil。

圖 3、Ingredients 使用個數統計



由圖 3 可以得知，大多數的 Cuisines 由 5-15 種 ingredients 組成。

7. Proposed Methods

這次我們使用的方法主要分為三個步驟，依序為：

1. 資料前處理
2. 基本演算法
3. 嘗試其他演算法與調整參數

7-1. 資料前處理

首先，先將對模型訓練無效的資料刪除，即刪除 ”id” 欄位。
再來，使用 is.null().sum() 檢查是否有遺失值，於訓練值與測試集均未發現遺失值。
接下來，因為這次模型的 input 與 output 的原始資料都是文字，因此，我們使用了以下兩種方式將文字資料轉換為可以投入模型的數值資料：

1. one-hot encoding：one-hot encoding 會將每個類別型變量，都建立一個新的二元變量，如果原始變量的取值為該類別，則將新變量的值設為 1，否則設為 0。

在程式中，我們使用了 scikit-learn 中的 OneHotEncoder，來將訓練、測試資料轉中的食材種類轉為二元型變量。

2. TF-IDF：TFIDF (term frequency–inverse document frequency) 是一種統計方法，用以評估單詞對於一份文件重要程度的衡量手法。
   - Term Frequency (TF) 詞頻：計算並找到出現次數最多的詞
   - Inverse Document Frequency (IDF) 逆文檔頻率：在詞頻的基礎上，對每個詞分配一個重要性的權重，最常見的詞給予最小的權重，較少見的詞給予較大的權重，也就是大小與一個詞的常見程度成反比。

   TF-IDF 的公式如下：$TF\text{-}IDF = TF \times IDF$
   其中，TF-IDF 值越高表示該詞越重要。

   在程式中，我們會先將 ”ingredients” 欄位的原始資料型態由 list 轉成 string，方法為使用 join() 將字詞加入字串中，並用空格做區隔。轉換為 string 的原因是希望讓 TF-IDF 認為此欄位內的資料是文本，進而算詞頻等。接著，從 sklearn 套件 import CountVectorizer 和 TfidfVectorizer，創建詞袋數據庫，並透過 CountVectorizer 中的 transform 函數將文本中的詞語先轉換成詞頻矩陣，再利用 TfidfVectorizer 建構一個計算 TF-IDF 的函式，依詞頻出現多寡給予各食材不同的權重，最終將詞頻矩陣轉換成 TF-IDF 權重矩陣。藉由上述的步驟，即可將文字資料轉為數字組成矩陣，方可以放進模型計算。

7-2. 基本演算法

在基本演算法中，我們分別就上述兩種資料前處理方式於 Logistic Regression、Decision Tree、Random Forest、SVM 等四種模型測試表現結果好壞，結果如下：

表 4、Baseline Models 表現結果

	Accuracy	
Model	**TF-IDF**	**One-Hot Encoding**
Logistic Regression	0.7856	0.1927
Decision Tree	0.6258	0.5413
Random Forest	0.7556	0.7098
SVM	0.7816	0.2358

【註：表 4 模型皆使用預設參數】

7-3. 嘗試進階演算法與調整參數

在經過四種 baseline methods 測試過後，可以發現經 one-hot encoding 後的表現明顯較差，因此，於後續測試進階模型如：XGBoost、LightGBM、Gradient Boosting 時，皆使用 TF-IDF 來做為資料前處理方法。

表 5、Advanced Models 表現結果

Model	**Accuracy**
XGBoost	0.7325
LightGBM	0.7764
Gradient Boosting	0.7387
Neural Network (2-layer)	0.6782

表 5 為經過 TF-IDF 處裡後，使用較進階的機器學習或神經網路方法，且使用預設參數的結果。

在測試過後，發現結果與競賽排行榜前幾名還有段距離，因此挑選表現較佳的 SVM 與 LightGBM 進行進一步調整參數，以追求更高的準確度。以下使調整參數方法皆為使用 GridSearchCV 來調整參數：

1. SVM 調整參數

   最終使用的參數如下：

   - C = 10
   - kernel = ‘rbf’（ Gaussian 高斯 ）
   - gamma = ‘scale’
   - decision_function_shape = ‘ovo’（ one-vs-one 一對一 ）

2. LightGBM 調整參數

   最終使用的參數如下：

   - learning_rate = 0.25
   - n_estimators = 1000
   - max_depth = 6

最終成果如下：

表 6、SVM 與 LightGBM 調整參數後表現結果

模型	預設參數	調整參數後
SVM	0.7816	0.8108
LightGBM	0.7764	0.7862

由表 6 可見，兩模型於調整參數後表現皆有顯著提升。

8. Experimental Results and Analysis

8.1 資料前處理方法分析

表 4、Baseline Models 表現結果

	Accuracy	
Model	**TF-IDF**	**One-Hot Encoding**
Logistic Regression	0.7856	0.1927
Decision Tree	0.6258	0.5413
Random Forest	0.7556	0.7098
SVM	0.7816	0.2358

由表 4 可見，相較 TF-IDF 的資料前處理方法，one-hot encoing 的表現遜色了不少，推測是因為將食材使用 one-hot encoing 後過於分散，只能表示此種料理有無這種食材，無法表示食材間有互相相關，因此預測準確率大幅下降，還徒增運算時間。此外，特別是於 Logistic Regression 與 SVM 兩種模型中，此兩種模型對 one-hot encoding 所生成的稀疏資料表現極差，推測是因為是這些模型在計算梯度和求解優化時，通常使用平方損失函數或 Hinge 損失函數，其中對應的偏微分方程式中含有變量的平方項。因此對於稀疏資料，這些平方項對模型的影響會非常小，模型就可能會忽略大多數變量，導致表現不佳。

8.2 模型於 TF-IDF 的表現

由初步模型分析結果可以得知，SVM 與 Logistic Regression 的表現結果最佳，甚至較其他進階模型 (如：LightGBM、XGBoost)表現好上不少，這個結果蠻出乎我們預料的，推測是因為此數據集經 TF-IDF 轉化過的特徵較不適何此進階模型，像是單詞出

現頻率沒有很好地反映文檔的類別，則使用基於樹的模型 (如： XGBoost 和 LGBM) 就可能無法很好地學習資料，導致模型表現不佳。另外，也可能是因為資料不平衡導致表現不佳，像由圖 3 可以發現某些類別的文檔數量遠少於其他類別，就會讓模型更加偏向多數類別，導致少數類別的表現較差。因此，後續若是使用 oversampling 或 undersampling 是有機會再提高其表現的。

8.3 LightGBM 調整參數

在調整 LightGBM 參數的過程中，藉機複習了各參數背後的邏輯，因此想在這裡分享一下條參數的歷程：

a. learning rate

表 7、LightGBM 調整 learning rate 的表現結果

Learning rate	Accuracy	Precision
0.001	0.3094	0.314
0.01	0.6777	0.6774
0.05	0.7612	0.7618
0.1	0.7764	0.7788
0.25	**0.7803**	**0.7785**

註：其他參數皆使用預設參數；註 2：表現最好之參數用綠色標示

learning rate 是指每次迭代更新模型參數時的步長大小。當 learning rate 較大時，模型的參數更新幅度較大，但也有可能跳過最佳解，使得模型的表現較差。反之，當 learning rate 較小時，模型的參數更新幅度較小，但也有可能花費較長的時間才能收斂到最佳解，使得訓練速度較慢。而在這次調整參數的過程中，learning rate=0.25 時有最佳解。

b. n_estimators

表 8、LightGBM 調整 n_estimators 的表現結果

n_estimators	Accuracy	Precision
10	0.7231	0.7206
50	0.7721	0.7721
100	0.7803	0.7785
500	0.7845	0.7828
1000	**0.7862**	**0.7853**

註：learning rate = 0.25，其餘參數皆使用預設參數

n_estimators 指的是使用的決策樹的數量。較大的 n_estimators 有助於提高模型的精度，但也有可能會使模型過於複雜，導致過擬合。而在這次調整參數的過程中，n_estimators = 1000 時有最佳解。

c. max_depth

表 9、LightGBM 調整 max_depth 的表現結果

max_depth	Accuracy	Precision
3	0.7234	0.7134
5	0.7348	0.7417
6	**0.7821**	**0.7778**
7	0.7652	0.7436

註：learning rate = 0.25、n_estimators = 1000，其餘參數皆使用預設參數

max_depth 指的是決策樹的最大深度。深度越大的決策樹可以對訓練資料進行更精確的建模，但也有可能會使模型過於複雜，導致過擬合。而在這次調整參數的過程中，max_depth = 6 時有最佳解。

9. Discussions

在這次的期末報告中，除了上述方法外，還測試過神經網路，原以為能使用 TF-IDF 加上 2 置 3 層的神經網路可以很好地搭建模型，不過測試的效果皆不盡理想，推測是因為 TF-IDF 自身的限制，因其背後的原理是以計算詞頻為本，很難表現出單詞之間的關聯性，因此可能無法將神經網路訓練起來。原本另外有打算嘗試 Bert，但想了一想資料文本中的食材雖看得出之間的關聯性，但看不出如一般文本單字間的「上下文關係」，因此猜測表現也不會到很好。就目前於網路上收集到的資訊，我認為此競賽是有準確率的侷限性的，由排行榜上最高準確率大約為 0.8300 也可以推斷得此結果。