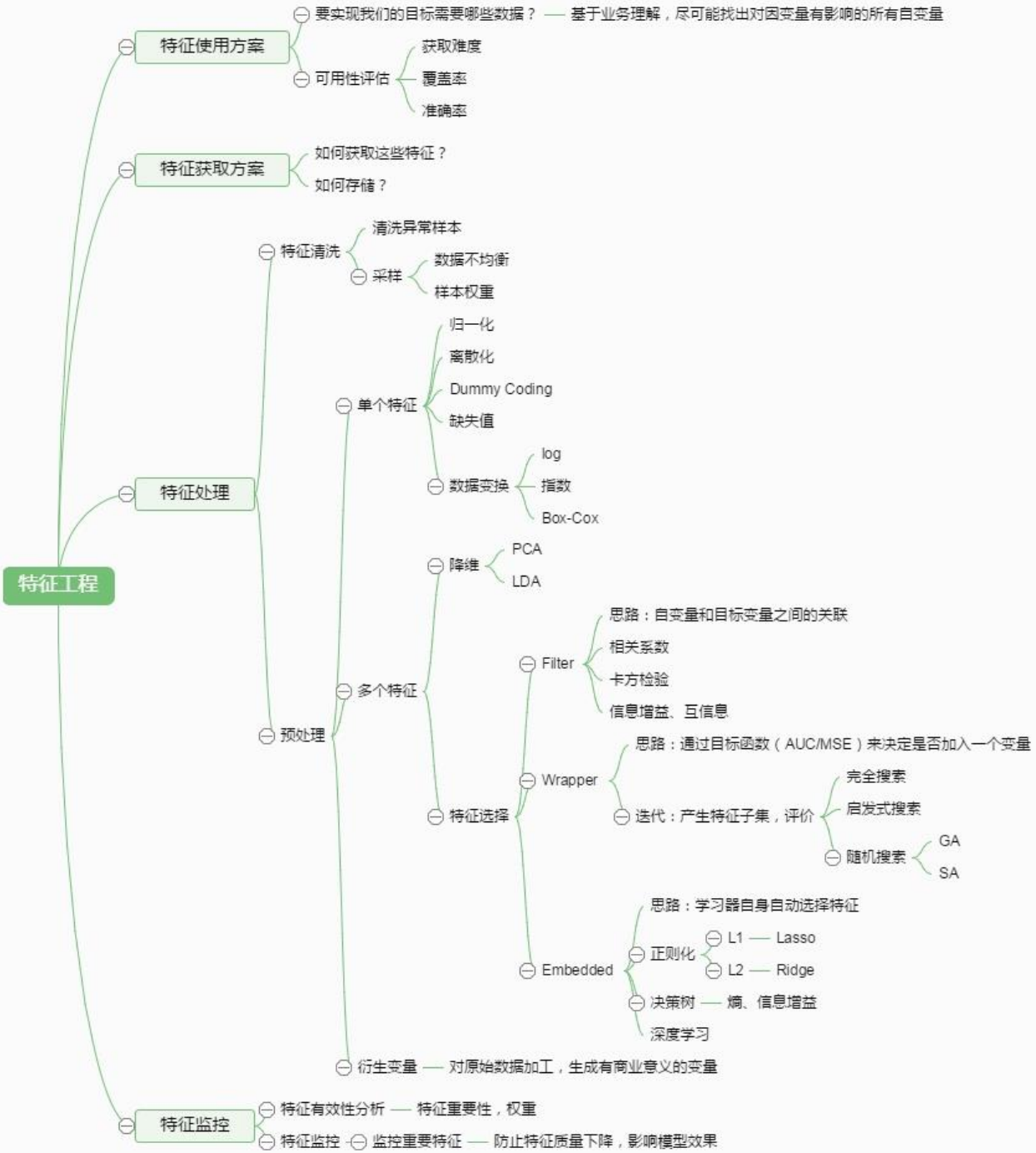
Feature Engineering
特徵工程

特徵工程是什麼？

- Feature Engineering 是把 raw data 轉換成 features 的整個過程的總稱。基本上特徵工程就是個手藝活，講求的是創造力。
- 有這麼一句話在業界廣泛流傳：「資料和特徵決定了機器學習的上限，而模型和演算法只是逼近這個上限而已。」那特徵工程到底是什麼呢？顧名思義，其本質是一項工程活動，目的是最大限度地從原始資料中提取特徵以供演算法和模型使用

什麼「不是」特徵工程？

- 最初的數據收集
- 創造目標變量
- 數據清洗 (移除重複變量、處理丟失的數據、修正遺失的標籤等工作)
- 規模化或者歸一化 (交叉驗證過程)
- 利用PCA選取特徵 (屬於交叉驗證過程)


特征工程
特征使用方案
要实现我们的目标需要哪些数据？ — 基于业务理解，尽可能找出对因变量有影响的所有自变量
可用性评估
获取难度
覆盖率
准确率
特征获取方案
如何获取这些特征？
如何存储？
特征处理
特征清洗
清洗异常样本
采样
数据不均衡
样本权重
预处理
单个特征
归一化
离散化
Dummy Coding
缺失值
数据变换
log
指数
Box-Cox
多个特征
降维
PCA
LDA
特征选择
Filter
思路：自变量和目标变量之间的关联
相关系数
卡方检验
信息增益、互信息
Wrapper
思路：通过目标函数（AUC/MSE）来决定是否加入一个变量
迭代：产生特征子集，评价
完全搜索
启发式搜索
随机搜索
GA
SA
Embedded
思路：学习器自身自动选择特征
正则化
L1 — Lasso
L2 — Ridge
决策树 — 熵、信息增益
深度学习
衍生变量 — 对原始数据加工，生成有商业意义的变量
特征监控
特征有效性分析 — 特征重要性，权重
特征监控 — 监控重要特征 — 防止特征质量下降，影响模型效果

- 在機器學習的實際應用中，特徵數量可能較多，其中可能存在不相關的特徵，特徵之間也可能存在相關性，容易導致如下的後果：
  - 特徵個數越多，分析特徵、訓練模型所需的時間就越長，模型也會越複雜。
  - 特徵個數越多，容易引起「維度災難」，其推廣能力會下降。
  - 特徵個數越多，容易導致機器學習中經常出現的特徵稀疏的問題，導致模型效果下降。
  - 對於模型來說，可能會導致不適定的情況，即是解出的參數會因為樣本的微小變化而出現大的波動。
- 特徵選擇，能剔除不相關、冗餘、沒有差異刻畫能力的特徵，從而達到減少特徵個數、減少訓練或者運行時間、提高模型精確度的作用。

特徵工程的重要性

- 特徵越好，靈活性越強
  - 只要特徵選得好，即使是一般的模型（或算法）也能獲得很好的性能，因為大多數模型（或算法）在好的數據特徵下表現的性能都還不錯。好特徵的靈活性在於它允許你選擇不複雜的模型，同時運行速度也更快，也更容易理解和維護。
- 特徵越好，構建的模型越簡單
  - 有了好的特徵，即便你的參數不是最優的，你的模型性能也能仍然會表現的很nice，所以你就不需要花太多的時間去尋找最有參數，這大大的降低了模型的複雜度，使模型趨於簡單。
- 特徵越好，模型的性能越出色
  - 顯然，這一點是毫無爭議的，我們進行特徵工程的最終目的就是提升模型的性能。

異常値處理

Missing Value Imputation

最簡單暴力的做法當然就是直接刪除掉那些含有缺失值的資料樣本。

針對 numerical 特徵的缺失值，可以用以下方式取代：

- 0，缺點是可能會混淆其他本來就是 0 的數值
- -999，用某個正常情況下不會出現的數值代替，但是選得不好可能會變成異常值，要特別對待
- Mean，平均數
- Median，中位數，跟平均數相比，不會被異常值干擾

針對 categorical 特徵的缺失值，可以用以下方式取代：

- Mode，眾數，最常見的值
- 改成 "Others" 之類的值

假設你要填補 age 這個特徵，然後你有其他例如 gender 這樣的特徵，你可以分別計算男性和女性的 age 的 mean、median 和 mode 來填補缺失值；更複雜一點的方式是，你可以把沒有缺失值的數據挑出來，用它們來訓練一個 regression 或 classification 模型，用這個模型來預測缺失值。

不過其實有些演算法是可以容許缺失值的，這時候可以新增一個 has_missing_value 欄位（稱為 NA indicator column）。

Outliers Detection

- 發現離群值最直觀的方式就是畫圖表，針對單一特徵可以使用 box plot；兩兩特徵則可以使用 scatter plot。
- 處置離群值的方式通常是直接刪除或是做變換(例如 log transformation 或 binning)，當然你也可以套用處理缺失值的方式。

Duplicate Entries Removal

- duplicate 或 redundant 尤其指的是那些 features 都一樣，但是 target variable 卻不同的數據。

特徵縮放
Feature Scaling

Standardization 標準化

- 原始資料中，因為各個特徵的含義和單位不同，每個特徵的取值範圍可能會差異很大。例如某個二元特徵的範圍是 0 或 1，另一個價格特徵的範圍可能是 [0, 1000000]，由於取值範圍相差過大導致了模型可能會更偏向於取值範圍較大的那個特徵。解決的辦法就是把各種不同 scale 的特徵轉換成同樣的 scale，稱為標準化或正規化。
- 狹義來說，標準化專門指的是透過計算 z-score，讓數據的 mean 為 0、 variance 為 1。
- **標準差標準化**
  - 經過處理的資料符合標準正態分佈，即均值為0，標準差為1，其轉化函數為：

$$x^* = \frac{x - \mu}{\sigma}$$

  - 其中μ為所有樣本資料的均值，σ為所有樣本資料的標準差

Normalization 歸一化、正規化

- 歸一化是指把每個樣本縮放到單位範數(每個樣本的範數為 1)，適用於計算 dot product 或者兩個樣本之間的相似性。除了標準化、歸一化之外，其他還有透過最大、最小值，把數據的範圍縮放到 [0, 1] 或 [-1, 1] 的區間縮放法，不過這個方法容易受異常值的影響。
- **線性歸一化**

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

  - 這種歸一化方法比較適用在數值比較集中的情況。這種方法有個缺陷，如果max和min不穩定，很容易使得歸一化結果不穩定，使得後續使用效果也不穩定。實際使用中可以用經驗常量來替代max和min

- **非線性歸一化**
  - 經常用在資料分化比較大的場景，有些數值很大，有些很小。通過一些數學函數，將原始值進行映射。該方法包括 log、指數，正切等。需要根據資料分佈的情況，決定非線性函數的曲線，比如log(V, 2)還是log(V, 10)等
  - **log10 函數轉換**
    - 通過以10為底的log函數轉換同樣可以實現歸一化，公式如下：

$$x = \frac{log_{10}(x)}{log_{10}(X_{max})}$$

    - 其中，Xmax是數據集中的最大值，並且數據集所有樣本的值都要大於等於1
  - **atan函數轉換**
    - 通過反正切函數實現歸一化公式如下：

$$x = atan(x) * \frac{2}{\pi}$$

    - 其中，當數據集都大於等於0時，映射空間在[0,1]之間，小於等於0時，映射空間在[-1,0]之間。

Logistic 函數轉換

$$f(x) = \frac{L}{1 + e^{-k(x - x_0)}}$$

e = the natural logarithm base
x_0 = the *x*-value of the sigmoid's midpoint,
L = the curve's maximum value, and
k = the logistic growth rate or steepness of the curve.[1]

$$f(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1} = \frac{1}{2} + \frac{1}{2}\tanh\left(\frac{x}{2}\right)$$

其中，當所有數據集中映射空間在 [0,1] 之間。

Z-score 函數轉換

$$z = \frac{x - \mu}{\sigma}$$

x 是需要被標準化的原始分數
μ 是母體的平均值
σ 是母體的標準差

標準分數（Standard Score，又稱z-score，中文稱為Z-分數或標準化值）在統計學中是一種無因次值，就是一種純數字標記，是藉由從單一（原始）分數中減去母體的平均值，再依照母體（母集合）的標準差分割成不同的差距，按照z值公式，各個樣本在經過轉換後，通常在正、負五到六之間不等。

小結

- 標準化是分別對單一特徵進行(針對column)；歸一化是對每個 Observation 進行(針對row)。
- 對 SVM、Logistic regression 或其他使用 Squared loss function 的演算法來說，需要 Standardization
- 對 Vector Space Model 來說，需要 Normalization
- 至於 Tree-based 的演算法，基本上都不需要標準化或歸一化，它們對 scale 不敏感。

特徵變換
Feature Transformation

指示器變量

特徵工程將要首先介紹的是用於分離關鍵信息的指示器變量。這個時候你也許會有疑問「難道不應該利用算法自己去學習處關鍵信息嘛？」然而事情並不是這樣的，這取決於我們擁有的數據量和競爭信號的強度。可以通過事先的強調來幫助算法聚焦於重要的特徵。

- **來自閾值的指示器變量：**例如我們研究美國消費者的酒精偏好時，數據集內包含著年齡特徵 age，這時候就可以創造一個指示器變量age>=21來區分達到法定飲酒年齡的人群。
- **來自多特徵的指示器變量：**你需要通過手上的兩個特徵 n_bedrooms 和 n_bathrooms 來預測房價，如果兩室兩衛的房產時可出租的優質資產，你就可以建立一個指示器變量來標記它。
- **針對特殊事件的指示器變量：**你正在為電子商務網站的每周銷量建模，這時候你就可以創造兩個指示器變量來表達黑色星期五購物節和聖誕節。
- **針對某一特徵組的指示器變量：**你正在進行網站轉化率的分析，數據集中包含分類特徵 traffic_source。你可以建立一個新的指示器變量 paid_traffic 來標記流浪來源價值的觀察結果，是 facebook 的廣告和還是 google 的廣告來的好呢？

適用於連續(Continuous)特徵：

- **Rounding**
  - 某些精度有到小數點後第 n 位的特徵，如果你其實不需要那麼精確，可以考慮 round(value * m) 或 round(log(value)) 這樣的做法，甚至可以把 round 之後的數值當成 categorical 特徵。

confidence	round(confidence * 10)
0.9594	10
0.1254	1
0.1854	2
0.5454	5
0.3655	4

適用於連續(Continuous)特徵：(續)

- **Log Transformation**
  - 因為 x 越大，log(x) 增長的速度就越慢，所以取 log 的意義是可以 compress 大數和 expand 小數，換句話說就是壓縮 "long tail" 和展開 "head"。假設 x 原本的範圍是 [100, 1000]，log(x, 10) 之後的範圍就變成 [2, 3] 了。也常常使用 log(1 + x) 或 log(x / (1 - x))。
  - 另外一種類似的做法是 square root 平方根或 cube root 立方根（可以用在負數）。

適用於連續(Continuous)特徵：(續)

- **Binarization 二值化**
  - 對數值型的數據設定一個 threshold，大於就賦值為 1、小於就賦值為 0。例如 score，如果你只關心「及格」或「不及格」，可以直接把成績對應到 1 (score >= 60) 和 0 (score < 60)。或是你要做啤酒銷量分析，你可以新增一個 age >= 18 的特徵來標示出已成年。
  - 你有一個 color 的 categorical 特徵，如果你不在乎實際上是什麼顏色的話，其實也可以改成 has_color。

適用於連續(Continuous)特徵：(續)

- **Binning，也稱為 Bucketization**
  - 以 age 這樣的特徵為例，你可以把所有年齡拆分成 n 段，0-20 歲、20-40 歲、40-60 歲等或是 0-18 歲、18-40 歲、40-70 歲等（等距或等量），然後把個別的年齡對應到某一段，假設 26 歲是對應到第二個 bucket，那新特徵的值就是 2。這種方式是人為地指定每個 bucket 的邊界值，還有另外一種拆分法是根據數據的分佈來拆，稱為 quantization 或 quantile binning，你只需要指定 bucket 的數量即可。
  - 同樣的概念應用到其他地方，可以把 datetime 特徵拆分成上午、中午、下午和晚上；如果是 categorical 特徵，則可以先 SELECT count() ... GROUP BY，然後把出現次數小於某個 threshold 的值改成 "Other" 之類的。或者是你有一個 occupation 特徵，如果你其實不需要非常準確的職業資訊的話，可以把 "Web Developer"、"iOS Developer" 或 "DBA" 這些個別的資料都改成 "Software Engineer"。
  - binarization 和 binning 都是對 continuous 特徵做 discretization 離散化，增強模型的非線性泛化能力。

適用於分類(Categorical)特徵：

- **Integer Encoding，也稱為 Label Encoding**
  - 把每個 category 對應到數字，一種做法是隨機對應到0, 1, 2, 3, 4等數字；另外一種做法是依照該值出現的頻率大小的順序來給值，例如最常出現的值給0依序給 1, 2, 3 等等。如果是針對一些在某種程度上有次序的Categorical特徵(稱為 Ordinal)，例如「鑽石會員」「白金會員」「黃金會員」「普通會員」，直接Mapping成數字可能沒什麼問題，但是如果是類似Color或City這樣的沒有明顯大小的特徵的話，還是用 One-Hot Encoding 比較合適。不過如果用的是Tree-based的演算法就無所謂了。
  - 有些Categorical特徵也可能會用數字表示(例如id)，跟Continuous 特徵的差別是，數值的差異或大小對Categorical特徵來說沒有太大的意義。

適用於分類(Categorical)特徵： (續)

- **One-hot Encoding (OHE)**
  - 如果某個特徵有 m 種值（例如 Taipei, Beijing, Tokyo），那它 one-hot encode 之後就會變成長度為 m 的向量：

```
city    city_Taipei city_Beijing city_tokyo
Taipei  1           0            0
Beijing 0           1            0
Tokyo   0           0            1
```

  - 你也可以改用 Dummy coding，這樣就只需要產生長度為 m -1 的向量：

```
city    city_Taipei city_Beijing
Taipei  1           0
Beijing 0           1
Tokyo   0           0
```

  - OHE 的缺點是容易造成特徵的維度大幅增加和沒辦法處理之前沒見過的值。

適用於分類(Categorical)特徵： (續)

- **Bin-counting**
  - 例如在 Computational Advertising 中，如果你有針對每個 user 的「廣告曝光數（包含點擊和未點擊）」和「廣告點擊數」，你就可以算出每個 user 的「點擊率」，然後用這個機率來表示每個 user，反之也可以對 ad id 使用類似的做法。

ad_id	ad_views	ad_clicks	ad_ctr
412533	18339	1355	0.074
423334	335	12	0.036
345664	1244	132	0.106
349833	35387	1244	0.035

  - 換個思路，如果你有一個 brand 的特徵，然後你可以從 user 的購買記錄中找出購買 A 品牌的人，有 70% 的人會購買 B 品牌、有 40% 的人會購買 C 品牌；購買 D 品牌的人，有 10% 的人會購買 A 品牌和 E 品牌，你可以每個品牌表示成這樣：

brand	A	B	C	D	E
A	1.0	0.7	0.4	0.0	0.0
B	...				
C	...				
D	0.1	0.0	0.0	1.0	0.1
E	...				

適用於分類(Categorical)特徵： (續)

- **LabelCount Encoding**
  - 類似 Bin-cunting 的做法，一樣是利用現有的 count 或其他統計上的資料，差別在於 LabelCount Encoding 最後用的是次序而不是數值本身。優點是對異常值不敏感。

ad_id	ad_clicks	ad_rank
412533	1355	1
423334	12	4
345664	132	3
349833	1244	2

適用於分類(Categorical)特徵： (續)

- **Count Vectorization**
  - 除了可以用在 text 特徵之外，如果你有comma-seperated的categorical特徵也可以使用這個方法。例如電影類型genre，裡頭的值長這樣 Action,Sci-Fi,Drama，就可以先用 RegexTokenizer 轉成 Array("action", "sci-fi", "drama")，再用 CountVectorizer 轉成 vector。。

適用於分類(Categorical)特徵： (續)

- **User Profile 用戶畫像**
  - 使用用戶畫像來表示每個 user id，例如用戶的年齡、性別、職業、收入、居住地、偏好的各種 tag 等，把每個 user 表示成一個 feature vector。除了單一維度的特徵之外，也可以建立「用戶聽過的歌都是哪些曲風」、「用戶（30 天內）瀏覽過的文章都是什麼分類，以 TF-IDF 的方式表達。或者是把用戶所有喜歡文章對應的向量的平均值作為此用戶的 profile。比如某個用戶經常關注與推薦系統有關的文章，那麼他的 profile 中 "CB"、"CF" 和 "推薦系統" 對應的權重值就會較高。

適用於分類(Categorical)特徵：(續)

- **Rare Categorical Variables**
  - 先計算好每一種 category 的數量，然後把小於某個 threshold 的 category 都改成 "Others" 之類的值。或是使用 clustering 演算法來達到同樣的目的。你也可以直接建立一個新的 binary feature 叫做 rare，要來標示那些相對少見的資料點。
- **Unseen Categorical Variables**
  - 當你用 **training set** 的資料 **fit**了一個 **StringIndexer(**和**OneHotEncoder)**，把它拿去用在**test set**上時，有一定的機率你會遇到某些 **categorical** 特徵的值只在 **test set** 出現，所以對只見過 **training set** 的 **transformer** 來說，這些就是所謂的 **unseen values**。
  - 對付 **unseen values** 通常有幾種做法：
    - 用整個 **training set + test set** 來編碼 **categorical** 特徵
    - 直接捨棄含有 **unseen values** 的那筆資料
    - 把 **unseen values** 改成 **"Others"** 之類的已知值。**StringIndexer** 的 **.setHandleInvalid("keep")** 基本上就是這種做法
- **Large Categorical Variables**
  - 針對那種非常大的 categorical 特徵（例如 id 類的特徵），如果你用的是 logistic regression，其實可以硬上 one-hot encoding。不然就是利用上面提到的 feature hashing 或 bin counting 等方式

特徵建構
Feature Construction

特徵建構及表示

特徵構建指的是從原有的特徵中，人工地創造出新的特徵，通常用來解決一般的線性模型沒辦法學到非線性特徵的問題。其中一個重點是能不能夠過某些辦法，在特徵中加入某些「額外的資訊」，因為你數據可能並非完全理想，你需要考慮是否可以利用不同的方式表示數據以獲得更多的信息，另外，也需要小心數據偏見的問題。

- 如果你有很多 user 購物的資料，除了可以 aggregate 得到 total spend 這樣的 feature 之外，也可以變換一下，變成 spend in last week、spend in last month 和 spend in last year 這種可以表示「趨勢」的特徵。

- **日期和時間特徵(Temporal Features)：**
  - 對於 date / time 類型的資料，除了轉換成 timestamp 和取出 day、month 和 year 做成新的欄位之外，也可以對 hour 做 binning(分成上午、中午、晚上之類的)或是對 day 做 binning(分成工作日、週末)；或是想辦法查出該日期當天的天氣、節日或活動等訊息，例如 is_national_holiday 或 has_sport_events。
  - 更進一步，用 datetime 類的資料通常也可以做成 spend_hours_last_week 或 spend_money_last_week 這種可以用來表示「趨勢」的特徵。
  - 或者購買日期和時間 (purchase_datetime) 會比一周內的購買日(purchase_day_of_week)和一天內的購買小時(purchase_hour_of_day)特徵更為有效，同時你還可以建立過去三十天的購買量(purchases_over_last_30_days)這樣的特徵來增強數據所能表達的信息
- **文字特徵 (Text Features)**
- **地理特徵(Spatial Features)**
  - 如果你有 city 或 address 等特徵，可以新建出 latitude 和 longitude 兩個 features（當然你得透過外部的 API 或資料來源才做得到），再組合出 median_income_within_2_miles 這樣的特徵。

外部數據

在特徵工程中外部數據很多情況下沒有被重充分利用，實際上它們可以為模型的性能帶來最重大的突破。例如定量對沖基金就通過各種不同金融數據源的層疊來進行研究。機器學習中的很多問題可以通過外部數據的引入得以解決，例如以下的一些實際應用：

- **時間序列數據**：在利用時間序列數據時你只需要類似日期的特徵就可以將不同的數據集進行疊加
- **外部API**：外部的API可以幫助我們得到更多的特徵，例如微軟的計算機視覺API就可以通過一張圖像返回很多人臉數據
- **地理編碼**：但我們擁有州、城市和接到地址的時候，你就可以將他們編碼成經緯度，這將有助於你在其他數據集的幫助下計算人工統計資料
- **異源的相同數據**：想像以下你能用多少種方式追蹤Facebook的廣告競價，可以利用facebook自集的追蹤工具、谷歌的分析結果以及其他第三方軟體，它們都會提供一些其他所不具有的信息。並且，數據間的任何差異都將為我們帶來更多的信息。

特徵交互
Features Interaction

特徵交互

接下來特徵工程中將要介紹兩個或者多個特徵間的交互：你記憶中有沒有聽過這樣的話：「整體優於部分」？實際上一些特徵可以結合起來提供比單個特徵更多的信息，甚至你還可以對他們進行加減乘除等操作來獲得更有效的特徵。

假設你有 A 和 B 兩個 continuous 特徵，你可以用 A + B、A - B、A * B 或 A / B 之類的方式建立新的特徵。

還有一種類似的作法叫 Polynomial Expansion 多項式展開，當 degree 為 2 時，可以把 (x, y) 兩個特徵變成 (x, x * x, y, x * y, y * y) 五個特徵。

- **註**：不建議使用自動循環來對所有特徵獲取相關的操作，這會導致「特徵爆炸」。
- **兩個特徵相加**：你想要通過預售數據來預測收入，通過 sales_blue_pens 和 sales_black_pens 相加，你可以得到 sales_pens 的總銷量
- **兩個特徵差異**：同樣也可以根據房屋的建造時間(house_built_date)和購買時間差(house_purchase_date)來得到房屋購買時的年限(house_age_at_purchase)
- **兩個特徵相乘**：當你要進行售價測試的時候，你可以通過售價price和指示器變量conversion相乘來得到新的特徵earnings。
- **兩個特徵相除**：當你在對市場競爭對手分析時，可以通過點擊率(n_clicks)和網頁打開次數(n_impressions)相除來得到點擊率click_through_rate來更好的分析對手數據。

特徵組合
Feature Combination

特徵組合，也稱為特徵交叉

- 特徵組合主要是針對 categorical 特徵，特徵交互則是適用於 continuous 特徵。但是兩者的概念是差不多的，就是把兩個以上的特徵透過某種方式結合在一起，變成新的特徵。通常用來解決一般的線性模型沒辦法學到非線性特徵的問題。
- 假設有 gender 和 wealth 兩個特徵，分別有 2 和 3 種取值，最簡單的方式就是直接 string concatenation 組合出一個新的特徵 gender_wealth，共有 2 x 3 = 6 種取值。因為是 categorical 特徵，可以直接對 gender_wealth 使用 StringIndexer 和 OneHotEncoder。你當然也可以一起組合 continuous 和 categorical 特徵，例如 age_wealth 這樣的特徵，只是 vector 裡的值就不是 0 1 而是 age 本身了。
- 假設 C 是 categorical 特徵，N 是 continuous 特徵，以下有幾種有意義的組合：

☐median(N) GROUP BY C 中位數
☐max(N) GROUP BY C 最大值
☐mean(N) GROUP BY C 算術平均數
☐std(N) GROUP BY C 標準差
☐mode(N) GROUP BY C 眾數
☐var(N) GROUP BY C 方差
☐min(N) GROUP BY C 最小值
☐N - median(N) GROUP BY C

user_id	age	gender	wealth	gender_wealth	gender_wealth_ohe	age_wealth
1	56	male	rich	male_rich	[1, 0, 0, 0, 0, 0]	[56, 0, 0]
2	30	male	middle	male_middle	[0, 1, 0, 0, 0, 0]	[0, 30, 0]
3	19	female	rich	female_rich	[0, 0, 0, 1, 0, 0]	[19, 0, 0]
4	62	female	poor	female_poor	[0, 0, 0, 0, 0, 1]	[0, 0, 62]
5	78	male	poor	male_poor	[0, 0, 1, 0, 0, 0]	[0, 0, 78]
6	34	female	middle	female_middle	[0, 0, 0, 0, 1, 0]	[0, 34, 0]

特徵選擇
Feature Selection

特徵選擇是指透過某些方法從所有的特徵中挑選出有用的特徵

- **Filter Method**
  - 採用某一種評估指標（發散性、相關性或 Information Gain 等），單獨地衡量個別特徵跟 target variable 之間的關係，常用的方法有 Chi Square Test（卡方檢驗）。這種特徵選擇方式沒有任何模型的參與。
  - 以相關性來說，也不見得跟 target variable 的相關性越高就越好
- **Wrapper Method**
  - 會採用某個模型來預測你的 target variable，把特徵選擇想成是一個組合優化的問題，想辦法找出一組特徵子集能夠讓模型的評估結果最好。缺點是太耗時間了，實務上不常用。
- **Embedded Method**
  - 通常會採用一個會為特徵賦予 coefficients 或 importances 的演算法，例如 Logistic Regression（特別是使用 L1 penalty）或 GBDT，直接用權重或重要性對所有特徵排序，然後取前 n 個作為特徵子集。

特徵學習
Feature Learning

- 也稱為 Representation Learning 或 Automated Feature Engineering。
  - GBDT
  - Neural Network: Restricted Boltzmann Machines
  - Deep Learning: Autoencoder

數據洩漏
Data Leakage

在 Features 中直接或間接地加入了跟 Target variable 有關的數據

錯誤分析（建模後處理）

特徵工程的最後一項內容我們稱之為建模結束後的錯誤分析。錯誤分析涉及廣泛的內容，包括誤分類的分析或者模型中觀察到的高錯誤率，通過這些來決定下一步的改進方向。

也許改進的方向包括收集更多的數據、拆分問題、針對誤差創造新的特徵。當利用誤差分析來進行特徵工程時，你需要理解為什麼你得模型會失效。下面舉一些具體的操作方法：

- **從大誤差開始**：誤差分析更多的是一個手工處理的過程，由於時間有限，我們建議你從大誤差的那些地方開始，尋找你可以建立新特徵的模式
- **類型分割**：另一種技術是將觀測值分割並分別與平均誤差進行比較，可以嘗試為誤差最高的部分建立指示器變量
- **非監督聚類**：當你在分類遇到問題時，可以在誤分類的觀測上運行非監督的聚類算法。我們不建議盲目地使用聚類作為新的特徵，但這會使得模式識別更為容易，自始至終都要記得我們的目標是找到為何會觀測到誤分類
- **向同事和領域專家諮詢**：上面三條都失效的情況下可以向同事和專家請教為何分類的表現如此糟糕。