資料前處理：

1. 缺失值填補
2. 離群值處理
3. 標準化

定義目標：

1. 回歸？分類？
2. 預測目標是什麼？
3. 用什麼資料預測？
4. Training set/ Validation set/ Test set

評估標準：

1. 回歸問題：
   1. RMSE, Root Mean Square Error
   2. Mean Absolute Error
   3. R-Square
2. 分類問題：
   1. Accuracy
   2. F1-score
   3. AUC, Area Under Curve

根據設定目標建立機器學習模型：

數據分析流程：

1. 收集數據

---

1. 數據清理
2. 特徵萃取
3. 資料視覺化>>>了解資料, 發現outliers 或異常數值, 分析個變數間的關練性
4. 建立模型
5. 驗證模型

---

1. 決策應用

Supervised Learning 監督學習

1. 前處理：
   1. 資料讀取
   2. 格式調整
      1. 資料類型
         1. Discrete variable: 整數單位
         2. Continuous variable
         3. 字串/類別轉換：
            1. Label encoding: 有序 (sklearn.preprocessing.LabelEncoder) Text

               Description automatically generated
            2. One hot encoding: 無序 (pd.get\_dummies)
      2. 特徵類型
         1. 數值型特徵
         2. 類別型特徵：一種類別對應一種分數
         3. 時間型特徵：週期性
   3. 填補缺值
      1. Median: np.median(value\_array)
      2. Quantiles: np.quantile(value\_arrar,q = ..)
      3. Mode: scipy.stats.mode(value\_array) or dictionary method
      4. Mean: np.mean(value\_array)
      5. 連續行數值標準化：
         1. Z 轉換
         2. 空間壓縮 (Ｙ＝0-1, -1~1）
         3. Regression model: 有差
         4. Tree-based model: 沒有太大關係
   4. 去離群值
      1. 透過scatter, histogram 檢查
      2. 透過describe() 檢查
      3. Emprical Cumulative Density Plot

A picture containing text

Description automatically generatedChart, line chart

Description automatically generated(009)

* + 1. 可捨棄或是調整 (需分析離群值之原因)
       1. 對數轉換 ln / log
       2. 縮尾 (將離群值換為 較中間的數)
       3. 截尾 （刪除）
       4. 插值 （賦予較合理之數）（隨機,均值,回歸等）＊＊＊
    2. 減少標準化/最小最大化（特徵縮放）之問題
  1. 特徵縮放
     1. Good:
        1. 對權重敏感或損失函數平滑程度有幫助者
        2. 特徵間的量級差異甚大
     2. Bad:
        1. 有些指標不適合在標準化空間進行
        2. 量的單位在某些特徵上有意義

1. Exploratory Data Analysis：  
   plt.style.use(‘default/ggplot/seaborn’)  
   統計值視覺化
   1. 相關係數
      1. Pd\_df.corr()[“Target”]
   2. 核密度函數KDE (Kernel Density Estimation)
      1. 線下面積和為1
      2. 對稱：K(-u) = K(u)
      3. 常用的Kernel function
         1. Gaussian (normal dist)
         2. Cosine
      4. 不同組間的分布差異
   3. 離散化
      1. 減少Outlier 對分析與模型的影響
      2. 等寬pd.cut(df,num\_group)
      3. 等頻pd.qcut(df,num\_group)
      4. 劇類
   4. 繪圖排版:subplot
      1. The index for subplot starts with 1
      2. 圖形必須直覺
      3. EDA讓人看懂資料
   5. 常用圖形
      1. Heatmap
         1. Sns.heatmap()
      2. Gridplot
         1. Sns.pairplot()
   6. 模型體驗
      1. 使用最簡單的模型當作baseline
2. 特徵工程:
   1. 從事實到對應分數的轉換
      1. 用於資料彙整之後,擬合模型之前
      2. 類別編碼
      3. 特徵縮放
   2. 數值型特徵：若離群值比例太高，或平均值不具代表性，執行去偏態
      1. 目標在於讓數值更接近常態分佈，讓平均值更具代表性
      2. 對數去偏：np.log1p()
      3. 方根去偏：減去min 後開根號 （sqrt）
      4. 分佈去偏(Boxcox):   
         from scipy import stats  
         stats.boxcox(df,lambda= λ)
         1. λ介於0~0.5之間
         2. df內之值必須>0
   3. 類別型特徵：
      1. 標籤編碼(Label Encoding)：單一col輸出
         1. 適合非深度學習模型
         2. 有序資料時可使用
         3. Pd.get\_dummies()
         4. Sklearn.preprocessing.OneHotEncoder()
      2. 獨熱編碼(One Hot Encoding)：多col輸出
         1. 適合深度學習模型
         2. 特徵重要性高，且可能值較少時使用
      3. 若特徵值與目標值有顯著相關：
         1. 均值編碼：容易overfitting，可參考出現頻率
         2. 平滑化：
         3. 計數編碼 (Counting)：類別筆數與目標值有高度相關，筆數可作為特徵
         4. 特徵雜湊 (Feature Hash)：鄉議類別的數量非常龐大時
   4. 時間型特徵：
      1. 時間特徵分解
         1. 年月日時分秒 週 星期
      2. 週期循環特徵：頭尾相接，需注意最高最低點的設置
         1. 週期：季, 月,日,年

Text, letter

Description automatically generated

* + 1. 時段特徵：短暫時段內的事件計數
       1. E.g.點擊網站前一天的累計點擊量等
  1. 特徵組合：機器學習關鍵在特徵工程，特徵工程關鍵在領域知識 （提升預測力）
     1. 合成特徵：由多個col產生
     2. 組合獨熱矢量：
        1. E.g. [behavior type X time of day] >>>狗吼叫 ＋時段
     3. 數值型：加減乘除
     4. 群聚編碼：類別與數值 （不易overfitting）
        1. 類別＋mean/min/max/count/etc e.g. 第三責任險 前一年保費平均為2500 共三筆
        2. Vs. 均值編碼Graphical user interface

           Description automatically generated
  2. 特徵篩選：
     1. 用領域知識挑選，並以特徵重要性選擇
     2. 寧濫勿缺
  3. 特徵評估：
     1. 特徵重要性Featrure Importance (樹狀模型)
        1. 決定分支的次數weight： (sklearn 僅有此方法)
        2. 特徵覆蓋度cover：覆蓋結果數量
        3. 損失函數降低量gain：loss function決定
     2. 排列重要性Permutation Importance
        1. 打散原始資料中單一特徵的排序，並套用於已訓練的模型，檢查預測準確率
  4. 葉編碼 ＋ 邏輯斯回歸
     1. Receiver Operating Characteristic Curve
        1. 越左上越好
        2. 可翻轉結果

1. 模型選擇
   1. 概論：
      1. 驗證基礎
         1. 資料切分Sklearn.model\_selection.train\_test\_split()
         2. K-fold Cross-validation: sklearn.model\_selection.KFold()
         3. sklearn.model\_selection.cross\_val\_score: cross\_val\_score(estimator, train\_X, train\_Y, cv=5).mean()
      2. 預測類型
         1. 回歸
            1. 可轉化為分類問題
         2. 分類
            1. 二分類Binary-class
            2. 多元分類Multi-class：一個樣本只能有一個分類
            3. 多標籤Multi-label：一個樣本可有多個分類
      3. 評估指標
         1. Accuracy:正確分類樣本數/總樣本數
         2. 回歸：
            1. Mean Absolute Error (MAE): [0, ∞]
            2. Mean Square Error (MSE): [0, ∞]
            3. R-square: [0,1]
         3. 分類：
            1. Area Under Curve (AUC): [0,1] 機率超過閥值判定為1
            2. F1-Score (Precision, Recall): [0,1]   
               2\*(Precision \* Recall) / (Precision + Recall)

Precision （精度）模型判定瑕疵,樣本確實為瑕疵的比例

Recall （召回率）模型判定瑕疵, 佔樣本所有瑕疵比例

* + - 1. 混淆矩陣 Confusion Matrix  
         Table

         Description automatically generated
  1. 基礎模型
     1. 線性回歸 Linear Regression：sklearn.linear\_model.LinearRegression
        1. 通常作為baseline參考
        2. 注意貢獻性與標準化限制
     2. 邏輯斯回歸Logistic Regression:sklearn.linear\_model.LogisticRegression
        1. 預測機率值
        2. Penalty:
           1. L1：模型較為稀疏，較快
           2. L2
        3. C: 正則化強度，數字越小，模型越簡單
        4. Solver:
           1. Newton-cg: L2 only / multiclass problems
           2. Lbfgs: L2 only / multiclass problems
           3. Liblinear:二分類問題或少量數據
           4. Sag: multiclass problems / fast for large one
           5. Saga: multiclass problems / fast for large one
        5. Multi-class:
           1. One-vs-rest: OvR, 都可
           2. Many-vs-many: MvM, 只能搭配newton-cg, sag
     3. 套索算法LASSO：Linear Regression + L1，可作feature selecting之用
        1. Sklearn.linear\_mode.Lasso
        2. Alpha:正則化強度
        3. .coef\_ : 印出訓練後的模型參數
     4. 嶺回歸 Ridge Regression：Linear Regression + L2，可解決多重共線性問題
        1. Sklearn.linear\_mode.Ridge
        2. Alpha:正則化強度
        3. .coef\_ : 印出訓練後的模型參數
  2. 樹狀模型
     1. 決策樹 Decision Tree: sklearn.tree.DecisionTreeClassifier/.DecisionTreeRegressor
        1. 使Information Gain 最大化
        2. 衡量資料相似度：
           1. 吉尼係數Gini-index
           2. 熵entropy
        3. .Feature\_importances\_:得知相對有用之feature為何
        4. 回歸決策數：返回range值e.g. 7, x<3 10, x>=3
        5. 缺點：容易overfitting
     2. 隨機森林 [sklearn.ensemble](https://scikit-learn.org/stable/modules/classes.html#module-sklearn.ensemble).RandomForestClassifier/.RandomForestRegressor
        1. 每棵樹隨機使用一部份訓練資料與特徵，每棵樹皆為獨立互不影響
        2. 多棵樹透過投票或加權得到最終結果
        3. 採用Bagging, 應用Bootstrapping思維 抽出又放回
        4. 缺點：訓練樣本小或特徵少時，效果不佳
        5. 特徵重要性會保留相同的特徵
     3. 梯度提升機 Gradient Boosting Machine

ensemble.GradientBoostingRegressor / .GradientBoostingClassifier

* + - 1. 利用後面生成的樹修正前面的樹，所以每棵樹皆有關聯
      2. 採用序列additive的方式產生
      3. 計算Gradient
      4. Y=score1\*learning\_rate+score2\*learning\_rae….
      5. 為回歸樹，調整後可做分類
      6. 特徵重要性可能會忽視相同的特徵

1. 參數調整
   1. 網格搜尋 Grid Search：
      1. 窮舉組合範圍
      2. 利用validation set 做最佳選擇參考
   2. 隨機搜尋 Random Search
      1. 用均勻分佈進行參數抽樣
      2. 通常結果較佳
   3. 先使用預設，再慢慢調整
   4. 超參數影響結果，但效果有限，data cleaning and feature engineering 更有效
   5. 分割train set 來做cross validation 選擇最佳參數組合確保泛化性，最後再用test 評估結果
2. 集成 Ensemble : 結合多個/多種分類器, 作為綜合預測。合議制/多數決
   1. 資料面集成：（不同資料，相同模型，合成結果）
      1. 裝袋法bagging：選出後放回, 如隨機森林
      2. 提升法 boosting ：由之前模型的預測結果，改變資料被抽到的權重或目標值。
         1. 自適應提升（AdaBoost, Adaptive Boosting）: 錯判資料被抽中機率放大，正確的縮小
         2. 梯度提升機 (Gradient Boosting Machine) ：依照殘差項調整新目標值
   2. 模型與特徵的集成：（相同資料，不同模型，合成結果）
      1. 混合泛化 Blending：前提是 單模效果好（有調參數）且模型差異大
      2. 投票泛化 Voting ：給予不同模型的預測值加權合成
      3. 堆疊泛化 Stacking ：用預測結果當新特徵。把模型當作下一階段的特徵編碼器使用

Unsupervised Learning 非監督學習：基於輸入資料找出模式

* 聚類分析：尋找隱藏模式
* 降低維度：縮減特徵維度
* 關聯法則,異常值偵測,探索性資料分析

1. 分群Cluster: 找出資料結構
   1. K-Mean
      1. 使總體裙內平方誤差最小
      2. 尋找k值
         1. 利用elbow method 找出最佳k值
         2. 計算Silhouette analysis 越接近1越好
      3. 分出k 不相似的clusters 並且cluster 內各點相似
      4. Initial設定不同，會得多不同結果。導致local optima 而非global optima
      5. 評估方式
         1. 有目標值：忽略目標值，後用結果與目標值側量準確性
         2. 無目標值：用資料本身的分布資訊
   2. Hierarchical Clustering階層分群法
      1. 由下而上,將距離最近的cluster 合併為一 ，直到剩下一個cluster
      2. 距離計算方式：
         1. Single-link: 不同群中最接近的兩點距離(大者恆大)
         2. Complete-link:不同群中最遠的兩點距離（齊頭並進）
         3. Average-link: 不同群間各點間距離總和平均
         4. Ward’s link:將兩群合併後，各點到合併後的群中心的距離平方和
      3. 不適合用在大資料
2. 降維 Dimension Deduction
   1. PCA Principle Components Analysis 主成分分析：
      1. 若資料有意義的維度太低，則幾個主要成分就可以將資料分解完畢
      2. 缺點：若特徵間是非線性關係，可能會underfitting, e.g.文字或影像資料
   2. T-SNE T分佈隨機近鄰嵌入：（數據可視化,了解和驗證數據或者模型）
      1. 缺點：運算費時
      2. 對於非線性特徵，有更好地表現
      3. 流行還原：在高維度到低維度的對應中，盡量保持資料點之間的遠近關係。
      4. 其他常見：Isomap，Locally Linear Embedding (LLE) 等
3. 深度學習 DNN（Supervised LearningDeep Neural Network）
   1. 簡介
      1. 類神經網 Neural Network
         1. 隱藏層數 1-2層
         2. 結構：感知器（Perceptron）+啟動函數 (Activation Function)
         3. 解決問題：
         4. 基礎回歸問題
      2. 深度學習 Deep Learning
         1. 隱藏層數 多層
         2. 結構：卷積神經網路（CNN）+遞歸神經網路（RNN）
         3. 影像、自然語言處理等多樣問題
      3. 觀察指標：已損失函數/誤差為主
      4. 概念
         1. CNN
            1. 目標：影像處理
            2. 神經連結精省
            3. DropOut 隨機移除
            4. BatchNormalization 批次正規化，有更好的傳導力
         2. RNN
            1. 目標：時序資料處理
            2. 時序間的橫向傳遞
         3. 結構
            1. 輸入層 Input Layer：第一層
            2. 隱藏層 Hidden Layer：最後一層
            3. 輸出層Output Layer：其他層
         4. 神經元：
            1. 啟動函數Activation Function：位於神經元內，將上一層神經元的輸入總和轉換成這一個神經元輸出的函數
            2. 損失函數 Loss Function：定義預測值與實際值的誤差大小
            3. 倒傳遞Back-Propagation：將損失值，轉化成類神經權重更新的方法，已達成各種應用的學習目標
      5. 重要概念
         1. 分類/回歸問題
         2. 遞迴次數 Epoch
         3. 神經元
         4. 訓練/測試誤差
         5. 學習曲線
         6. 隱藏層
         7. 特徵
         8. 批次大小 Batch Size：批次小，計算久
         9. 學習速率 Learning Rate: 速率大，收斂快。過大有可能無法收斂
         10. 正則化 Regularization
         11. 啟動函數：Tanh/Sigmoid/ReLu/Linear
   2. 套件介紹
      1. 內建dataset: <https://keras.io/api/datasets/>
      2. 如何搭建神經網路
         1. 序列模型 Sequential，單輸入，單輸出模型
         2. From keras.models import Sequential
         3. From keras.layers import Dense, Activation
         4. 建構：
            1. Model = Sequential([Dense(32,\_input\_shap=(784,)),Activation(“relu”)
            2. Model = Sequential()  
               Model.add(Dense(32,\_dim=784))  
               Model.add(Activation(“relu”))
         5. 模型配置：
            1. Model.compile
            2. Loss / optimizer/ metrics (評估指標)
         6. 模型訓練參數設置＋訓練 get parameters to the correct value to map inputs to outputs
            1. Model.fit
         7. 模型預測 predict
         8. 常用參數
            1. Dense
            2. Activation
            3. Dropout：對上層應用dropout防止overfitting
            4. Flatten: 對上層輸出一維化
            5. Reshape 對上層輸出reshape
         9. [函數式API](https://clownote.github.io/2020/08/17/DeepLearningWithPython/Deep-Learning%20with-Python-ch7_1/): 可重用訓練好的模型 （包括結構與權重）  
            1. 用以定義複雜模型，即多輸入 多輸出模型
            2. 把層當作函數使用，接收張量並返回張量
            3. 建構

Main\_input (Input Layer)

Hidden\_layer

Main\_output (Dense)

* 1. 組成概念：
     1. MLP 多層感知器：可用於非線性近似資料，進行分類或回歸運算，為監督式學習演算法
        1. 向前傳遞類神經網路
        2. 包含輸入層，隱藏層，輸出層
        3. 用倒傳遞，達到監督學習
        4. 缺點：對特徵預先處理很敏感，初始權重會影響驗證時準確率的浮動，需調整每層神經元數量、層數、迭代次數
     2. 目標函數：需要最大化 或 最小化的一個函數，稱為目標函數，用以衡量模型預測能力好壞
        1. 分為 分類問題損失函數 與 回歸問題損失函數
        2. 最小化：損失函數，即是實際值(y)與預測值y- ŷ的落差
           1. 回歸問題：殘差 residual
           2. 分類問題：錯誤率 error rate
           3. 均方誤差 MSE: 差距平均值
           4. 交叉熵Cross entropy(分類問題，兩個概率分佈之間的距離)
           5. Hinge Error
     3. 啟動函數：定義了每個節點的輸出和輸入關係的函數，提供非線性化能力
        1. Relu:
           1. 隱藏層
           2. 函數小於0時，輸出0；大於0時則為線性函數，輸出值等於輸入值
           3. 替代：

LeakyRelu

PRelu 可避免神經元死亡

Maxout

* + - 1. Sigmoid: 二分類 選sigmoid
         1. 輸出層
         2. 輸出限定在0-1之間，函數<0 輸出0, 函數>0 輸出1, 函數介於0-1 輸出 ＝輸入
         3. 傳遞過程不容易發散
         4. 缺點：Sigmoid函數在微分最大值只有1/4，只需再多幾次的值相乘，就會很接近0容易梯度消失（僅能做淺層傳播）。且 輸出值皆不是0
         5. 求cross entropy loss
      2. Tanh: 二分類, 幾乎所有場合都能用
         1. 輸出層
         2. 輸出值 -1 ~ 1 之間
         3. 在循環過程中不斷擴大特徵效果
      3. Softmax: 多元分類
         1. 輸出層
         2. 將k維向量，轉為由0-1之間數組成的向量，總和為1
         3. 作為多分類的機率預測
         4. 求cross entropy loss
         5. DNN分類時, 通常用於輸出層
    1. 倒傳遞：
       1. Gradient Descent 梯度下降 ：優化算法, 使模型收斂
          1. 通常開始時，使用較大的learning rate 然後逐漸縮小
          2. 梯度：將多個參數的偏倒數以向量的形式寫出來就是梯度
          3. Learning rate 參數：

Decayed\_learning\_rate 衰減後的學習率

Learning\_rate 初始學習率

Decay\_rate 衰減率

Global\_step 當前的step

Decay\_steps 衰減週期

decayed\_learning\_rate=learning\_rate\*decay\_rate^(global\_step/decay\_steps)

* + - * 1. momentum梯度下降加速:
        2. 若 momentum 係數β 與這一次的負梯度放相相同，下次的幅度就會加大，加速收斂
      1. Backpropagation反向傳播：用來更新權值以最小化損失函數。
    1. 優化器 Optimizer:通過改善訓練方式，更新網路參數，來最佳化損失函數，
       1. 梯度下降 Gradient Descent
       2. 隨機梯度下降法 SGD (針對大數據集)
          1. 速度快
          2. 震盪大
          3. Epoch 完整訓練次數
          4. Batchsize 批量大小 2的冪次方(2的次方數)
          5. Iteration 使用batchsize個樣本訓練一次
          6. 通常搭配Momentum
       3. Adagrad
          1. 不需要手動調整學習率
          2. 缺點：學習率最終會變得非常小
          3. Decay 學習率衰減值
       4. RMSprop ：適合非平穩目標 和 RNN
          1. 不用手動調整
          2. 可微調每個參數選不同學習率
          3. 不易出現learning rate快速下降
       5. Adam : 適合稀疏梯度和非平穩目標 (通常Adam最好)
          1. 容易收斂至局部最優
          2. 自動調整learning rate
          3. 校正方式考量一、二階矩陣，更加平穩
    2. 訓練技巧：
       1. 使用GPU與否
       2. Xs 是否適當標準化：盡量使用function
       3. Ys 是否適當處理 如onehot encoded, 盡量使用function
       4. Model 是否如預期架構
       5. 超參數 Hyper-Parameters 是否得 ：集中管理
       6. 防止過擬合Overfitting
          1. 訓練上表現遠大於驗證
          2. 可使用k-fold，增加樣本，減少參數，加入regularization
       7. 學習率Learning Rate調整
          1. Momentum: 在損失方向增加動量（類似慣性），協助擺脫平原或小山谷

Momentum: 在更新方向以外，加上一個固定向量，之時移動方向會介於gradient step 跟 momentum step之間

Nesterov Momentum: 拔草測風向：將momentum 納入 gradient計算中

* + - * 1. 學習率過大：無法收斂到更低的損失平面
        2. 學習率過小：損失改變幅度小，平原區無法找到正確方向
        3. Weight decay :對cost function 做正則化
        4. Learning rate decay: 減少學習率，提高尋優能力
      1. 正規化Regularization :加在layer中
         1. From tensorflow.keras.regularizers import l1
         2. Kernel\_regularizer=l1(0.001)(input\_layer)
      2. 隨機移除 Drop out: 隨機拿掉一些連結（weights =0）減低複雜度
         1. 刻意提升訓練難度
         2. 強迫模型的每個參數都有更前的泛化能力
      3. 批次標準化 Batch Normalization: 在每一層對每個特徵做標準化，最終得到每個特徵的均值為0，方差為1的分佈
         1. 可解決Gradient Vanishing 梯度消失問題
         2. 可以用較大的learning rate
         3. 加速訓練
         4. 能取代dropout 與 regularizes
      4. 客製化損失函數 Customized Loss Function
         1. 需使用tensor operations
         2. 需輸出tensor
         3. Def 自訂函數，用tensor operations 運算
         4. 自訂函數，至少需要有y\_true, & y\_pred
         5. 將自定義函數，置入compile 中
      5. 回呼 Callback：在overfitting前停止，阻止模型變得更糟  
         callbacks=[es, mc] 即可套用多個 如earlystop+ modelcheckpoint
         1. Callback and Early Stop

From tensorflow.keras.callbacks import EarlyStopping

Earlystop( )設定：

Default: monitor='val\_loss', min\_delta=0, patience=0, verbose=0, mode='auto', baseline=None, restore\_best\_weights=False)

Monitor: based on what score

Patience: epochs to wait

Verbose

Mode: min / max / auto

Min\_delta: 增大或減小的閥域，符合才算improvement

Baseline: 訓練會停止，如果在baseline之前沒有進步

restore\_best\_weights: 是否从具有监测数量的最佳值的时期恢复模型权重。 如果为 False，则使用在训练的最后一步获得的模型权重。

在model.fit中增加callbacks參數

* + - * 1. Model Check Point: 自動將目前最佳的模型權重存下

From tensorflow.keras.callbacks import ModelCheckpoint

Checkpoint()設定：

filepath: path to save

monitor: based on what score

verbose

mode: min / max /auto

save\_best\_only: if False, 保持整個模型，True，僅保存權重

在model.fit() 中增加在callbacks參數

* + - * 1. Reduce Learning Rate將learning rate 逐漸減小

ReduceLROnPlateau:經過n次epoch後，若performance沒進步，調降一次

From tensorflow.keras.callbacks import ReduceLROnPlateau

Factor: decreace rate

Min\_lr: the lower bond of learning rate

Monitor

Mode: min / max /auto

Cooldown: number of epochs to wait before resuming normal operation after lr has been reduced.

Patience

verbose

Schedule decay: 每n次調降一次

tensorflow.keras.callbacks import LearningRateScheduler

需設定scheduler的規則  
def scheduler(epoch, lr):  
 if epoch < 10:  
 return lr  
 else:  
 return lr \* tf.math.exp(-0.1)

在model.fit() 中增加在callbacks參數

* + - * 1. 自定義：on\_位置\_時段

位置

Batch

Train

Epoch

時段

Begin

End

僅需啟動想要的部分

方法

From tensorflow.keras.callbacks import Callback

寫一個class 去繼承callback的class

針對位置與時段進行步驟的定義

* + 1. 應用案例：
       1. 傳統方法
          1. 處理影像辨識
          2. 將影像素量化為特徵，透過機器學習來做分類或回歸
          3. 將RGB轉為顏色直方圖
       2. 進行CIFAR-10分類
          1. Hog 的調用:

檢測局部形狀信息

* + - * 1. Svm (support vector machine)在 opencv 的調用: maximum margin classifier :最佳分類線（超平面）
  1. 卷積神經網路 Convolutional Neural Network
     1. 簡介
        1. 卷積是數學乘法與加法
        2. 利用濾波器 filter 對圖像做卷積來尋找規則
        3. 尋找與濾波器有相同特徵的區域
        4. 輸出為特徵圖
        5. 特徵圖經 Relu activation function 輸出為特徵值供下層使用
        6. 濾波器就是CNN模型中的參數，由CNN做動從訓練資料中找出是和的濾波器
        7. Pixel 值 0-255 由黑到白
        8. Vs. DNN
           1. DNN (Deep Neural Network)會忽略資料形狀，因為輸入為一維資料
           2. CNN 有 Convolutional Layer卷積層與 Pooling Layer池化層來維持形狀，並且避免參數大幅增加
        9. 卷積：
           1. Sliding window抓出的值與 濾波器 filter做運算，並產出較原輸入圖小的特徵圖（維度較小）
           2. 最後輸出須由Relu activate function 產出
        10. 池化層：
            1. 將輸入圖尺寸縮小（通常為一半）以減少每張feature map的維度並保留重要特徵
            2. 最終輸出不由Relu activite function 產出
            3. 用途：

減少後續需要layer層數

抗干擾

減少過度擬合

* + - * 1. 平坦層 Flatten:

將n維資料轉為1維

將特徵資訊丟到Full connected layer 進行分類

各連結權重，在同一層中共享

* + - * 1. 全連接層 Fully connected layer:

將平坦層之輸出，丟入DNN做運算

Dense

Activation

* + 1. 套件練習
       1. Convolution:
          1. Filters: 卷積內核總數
          2. Input\_shape:僅第一次卷積需指定
          3. Kernel size: filter 尺寸
          4. Padding: 當餘下窗口不足是否加一圈0值的pixel , valid （捨棄）/ same（填入0,達成輸入與輸出大小一樣）
          5. Stride: Sliding window 移動步數
          6. 類型：

separable\_conv: 參數可大幅減少，硬體要求低

對影像做兩次卷積

(1)DetphWise Conv: 對影像的三個通道獨立卷積

(2)PointWise Conv:使用1\*1 filter 做卷積

參數: depth\_multiplier: 在depthwise conv時，輸出的各通道各feature map 數量

conv2d/ conv3d

* + - 1. Pooling:
         1. Pool\_size: (垂直，水平)方向縮小比例的因數。E.g. （2,2）會把張量的兩個維度都縮小一半，無條件捨去
         2. Strides:
         3. Padding
         4. Data\_format:維度輸入順序 channels\_last / channels\_first
         5. 類型：

Max\_pool: 最大池化，取最大值，保留紋理訊息

Avg\_pool:平均池化，取平均值，保留背景訊息

* + - * 1. 誤差：

鄰域大小受限造成的估計值方差增大( avg\_pool 可減少)

kernal超參數與內核造成估計均值的偏移( max\_pool)可減小

* + 1. 訓練技巧
       1. 處理大量數據 (model.fit\_generator)
          1. 批次讀取（batch）:一次讀取一批，直至訓練結束，釋出記憶體後再讀取第二批
          2. 使用 generator 完成：自動吐出下一個

將return 改成 yield

* + - * 1. 改成採用 .fit\_generator 訓練資料 (而不是.fit)
      1. 處理小量數據 (model.fitgenerator)
         1. 資料量不足
         2. 可使用 data augmentation：將資料進行隨機處理，如翻轉平移旋轉，改亮度等 （需在圖片標準化之前完成)
         3. 需分出train/validation set 再做增強
         4. From tensorflow.keras.preprocessing.image import ImageDataGenerator

Rotation\_range：旋轉

Width\_shift\_range：左右平移

Height\_shift\_range：上下平移

Brightness\_range：亮度調整

Zoom\_range：影像縮放

horizontal\_flip 或者 vertical\_flip: T/F

* + - * 1. ImageDataGenerator.flow(images, shuffle=T/F)) 進行影像進行增強
      1. 遷移學習(Transfer Learning)
         1. 保留convolutional base
         2. 僅訓練 classifier
         3. From tensorflow.keras.applications.resnet50 import ResNet50 套用imagenet訓練好的ResNet50網路結構
         4. 參數

Input\_shape(224,224,3)

Weights=’imagenet’ 使用訓練的權重做遷移學習

Pooling=’avg’ 使用global averatge pooling 把featuremaps 變成一維向量

Include\_top=False 移除最後的denselayer 因為要做自己的分類器

Trainable : 是否凍結參數，使之不得修改 T/F

* + - * 1. Resent\_model.output 取出最後一層的featuremaps 再flatten 以及dense上分類器
    1. 電腦視覺

EDA – 統計量化的方式

1. 集中趨勢：
   1. 平均值 (mean)
   2. 中位數 (median)
   3. 眾數 (mode)
2. 資料分散程度：
   1. Min
   2. Max
   3. Range
   4. Quartiles
   5. Variance
   6. Standard Deviation
3. pd.hist(bin=n)  
   Chart

   Description automatically generated
4. pd. value\_counts().plot(kind="bar")

Chart, waterfall chart

Description automatically generated

模型選擇流程：

1. 定義模型
2. 評估模型好壞
   1. 由目標函數objective function/損失函數loss function 來衡量
   2. 線性回歸：MSE (Mean Square error 均方差)越小越好
3. 找出最佳模型參數
   1. 梯度下Gradient Descent/增量訓練Additive Training
   2. Overfitting : 學習到噪音，反而降低實際預測效果
4. Overfitting:
   1. 增加資料量
   2. 降低模型複雜度
   3. 使用正則化(Regularization)：函數=損失函數＋正則化
5. Underfitting:
   1. 增加複雜度
   2. 減輕或不使用正則化

重載模型：

Tesnorflow.keras.models.load\_model(path\_to\_the\_model)

僅讀取權重

Tensorflow.kears.models.load\_weights(path\_to\_the\_weights\_of\_the\_model)

CNN 視覺化

<https://github.com/utkuozbulak/pytorch-cnn-visualizations>

JavaScript

https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Asynchronous/Async\_await