電子商務技術作業 8

第一部份:**Python**

載入「income.csv」(預測薪水是否能超過 50K)，請將資料切成訓練集與測試集

(random_state=15, test_size=0.3)並做必要的前處理。

1. 建立 KNN 模型，印出模型對訓練集、測試集的 Accuracy。(5%)

```
[19]: import pandas as pd # 引用套件並縮寫為 pd
      import numpy as np
      from sklearn import preprocessing
      import matplotlib.pyplot as plt
      from sklearn.model_selection import train_test_split
      pd.set_option('display.width',1000)
      df = pd.read_csv('/Users/xwlee/Desktop/ECt/hw8/income.csv')

[20]: from sklearn.preprocessing import LabelEncoder
      labelencoder = LabelEncoder()
      df['workclass'] = labelencoder.fit_transform(df['workclass'])
      df['education'] = labelencoder.fit_transform(df['workclass'])
      df['marital-status'] = labelencoder.fit_transform(df['workclass'])
      df['occupation'] = labelencoder.fit_transform(df['workclass'])
      df['relationship'] = labelencoder.fit_transform(df['workclass'])
      df['race'] = labelencoder.fit_transform(df['workclass'])
      df['gender'] = labelencoder.fit_transform(df['workclass'])
      df['native-country'] = labelencoder.fit_transform(df['workclass'])

[21]: from sklearn.model_selection import train_test_split
      from sklearn.neighbors import KNeighborsClassifier
      from sklearn import metrics
      x=df[['age','workclass','fnlwgt','education','educational-num','marital-status','occupation','relationship','race','gender','capital-gain','
      y=df[['income']]
      x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=15)
      knn = KNeighborsClassifier()
      knn.fit(x_train,y_train.values.ravel())
      y_test_predicted = knn.predict(x_test)
      accuracy = metrics.accuracy_score(y_test, y_test_predicted)
      print(accuracy)
      0.6230348598769652
```

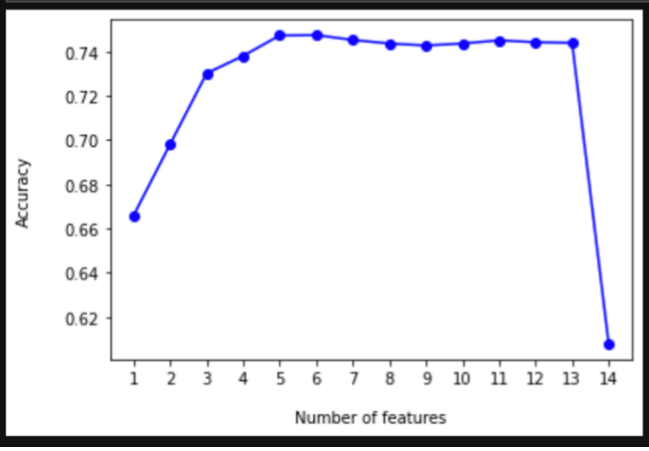
2. (承上題) 當限制 feature 數量為 n 時 (n = [1, 14])，請為每一個 n 利用 Wrapper feature selection 技巧尋找最適合此模型的 feature set。 搜尋策略請使用 forward selection， 並以 5 Cross Validation 後的 Accuracy 為選擇的依據。(30%)

```
[22]: from mlxtend.feature_selection import SequentialFeatureSelector as SFS
      a=[]
      for i in range(14):
          forward = SFS(knn,
                        k_features = i+1,
                        forward = True,
                        floating = False,
                        verbose = 0,
                        scoring = 'accuracy',
                        cv=5)
          forward = forward.fit(x_train, y_train.values.ravel())
          a.append(forward.k_score_)
          print("n=",i+1,f"Best score achieved: {forward.k_score_}, Feature's names: {forward.k_feature_names_}")
```

```
n= 1 Best score achieved: 0.6658819365115118, Feature's names: ('educational-num',)
n= 2 Best score achieved: 0.698117334105621, Feature's names: ('educational-num', 'capital-gain')
n= 3 Best score achieved: 0.7301902294874477, Feature's names: ('age', 'educational-num', 'capital-gain')
n= 4 Best score achieved: 0.738101137515486, Feature's names: ('age', 'educational-num', 'capital-gain', 'capital-loss')
n= 5 Best score achieved: 0.7473293324537834, Feature's names: ('age', 'educational-num', 'capital-gain', 'capital-loss', 'hours-per-week')
n= 6 Best score achieved: 0.7474764961734215, Feature's names: ('age', 'workclass', 'educational-num', 'capital-gain', 'capital-loss', 'hour
s-per-week')
n= 7 Best score achieved: 0.7452789084999919, Feature's names: ('age', 'workclass', 'education', 'educational-num', 'capital-gain', 'capital
-loss', 'hours-per-week')
n= 8 Best score achieved: 0.7436676159370157, Feature's names: ('age', 'workclass', 'education', 'educational-num', 'marital-status', 'capit
al-gain', 'capital-loss', 'hours-per-week')
n= 9 Best score achieved: 0.7427889241066401, Feature's names: ('age', 'workclass', 'education', 'educational-num', 'marital-status', 'occup
ation', 'capital-gain', 'capital-loss', 'hours-per-week')
n= 10 Best score achieved: 0.7436675086748293, Feature's names: ('age', 'workclass', 'education', 'educational-num', 'marital-status', 'occu
pation', 'relationship', 'capital-gain', 'capital-loss', 'hours-per-week')
n= 11 Best score achieved: 0.7451323883534717, Feature's names: ('age', 'workclass', 'education', 'educational-num', 'marital-status', 'occu
pation', 'relationship', 'race', 'capital-gain', 'capital-loss', 'hours-per-week')
n= 12 Best score achieved: 0.7442534819987235, Feature's names: ('age', 'workclass', 'education', 'educational-num', 'marital-status', 'occu
pation', 'relationship', 'race', 'gender', 'capital-gain', 'capital-loss', 'hours-per-week')
n= 13 Best score achieved: 0.7439601199191243, Feature's names: ('age', 'workclass', 'education', 'educational-num', 'marital-status', 'occu
pation', 'relationship', 'race', 'gender', 'capital-gain', 'capital-loss', 'hours-per-week', 'native-country')
n= 14 Best score achieved: 0.6078816254511715, Feature's names: ('age', 'workclass', 'fnlwgt', 'education', 'educational-num', 'marital-stat
us', 'occupation', 'relationship', 'race', 'gender', 'capital-gain', 'capital-loss', 'hours-per-week', 'native-country')
```

3. (承上題) 請以折線圖呈現模型在不同 **feature** 數量下的最佳表現。(X 軸:**feature** 數量，Y 軸 **Accuracy**)。(5%)

```
[23]: b = np.linspace(1,14,14)
      plt.plot(b,a,'o-',color = 'b')
      plt.xlabel("Number of features", fontsize=10, labelpad = 15)
      plt.ylabel("Accuracy", fontsize=10, labelpad = 20)
      plt.xticks(np.arange(min(b), max(b)+1, 1.0))
      plt.show()
```



4. 請印出最適合此模型的 **feature set**。(包含 **feature** 數量及名稱) **(5%)**

```
[24]: dict = {"Number of features": b,"Accuracy": a}
      df1 = pd.DataFrame(dict)
      df1
```

[24]:

	Number of features	Accuracy
0	1.0	0.665882
1	2.0	0.698117
2	3.0	0.730190
3	4.0	0.738101
4	5.0	0.747329
5	6.0	0.747476
6	7.0	0.745279
7	8.0	0.743668
8	9.0	0.742789
9	10.0	0.743668
10	11.0	0.745132
11	12.0	0.744253
12	13.0	0.743960
13	14.0	0.607882

```
[28]: n=df1['Number of features'].loc[df1['Accuracy'].argmax()]
      m=int(n.tolist())
      forward = SFS(knn,
                    k_features =m,
                    forward = True,
                    floating = False,
                    verbose = 0,
                    scoring = 'accuracy',
                    cv=5)
      forward = forward.fit(x_train, y_train.values.ravel())
      a.append(forward.k_score_)
      print("When number of feature is",m,f",best score achieved: {forward.k_score_}, Feature's names: {forward.k_feature_names_}")

      When number of feature is 6 ,best score achieved: 0.7474764961734215, Feature's names: ('age', 'workclass', 'educational-num', 'capital-gain
      ', 'capital-loss', 'hours-per-week')
```

5. 請使用挑選出的最佳 **features** 重新訓練模型 **(10%)**，並比較挑選前與挑選後的模型表現。**(10%)**

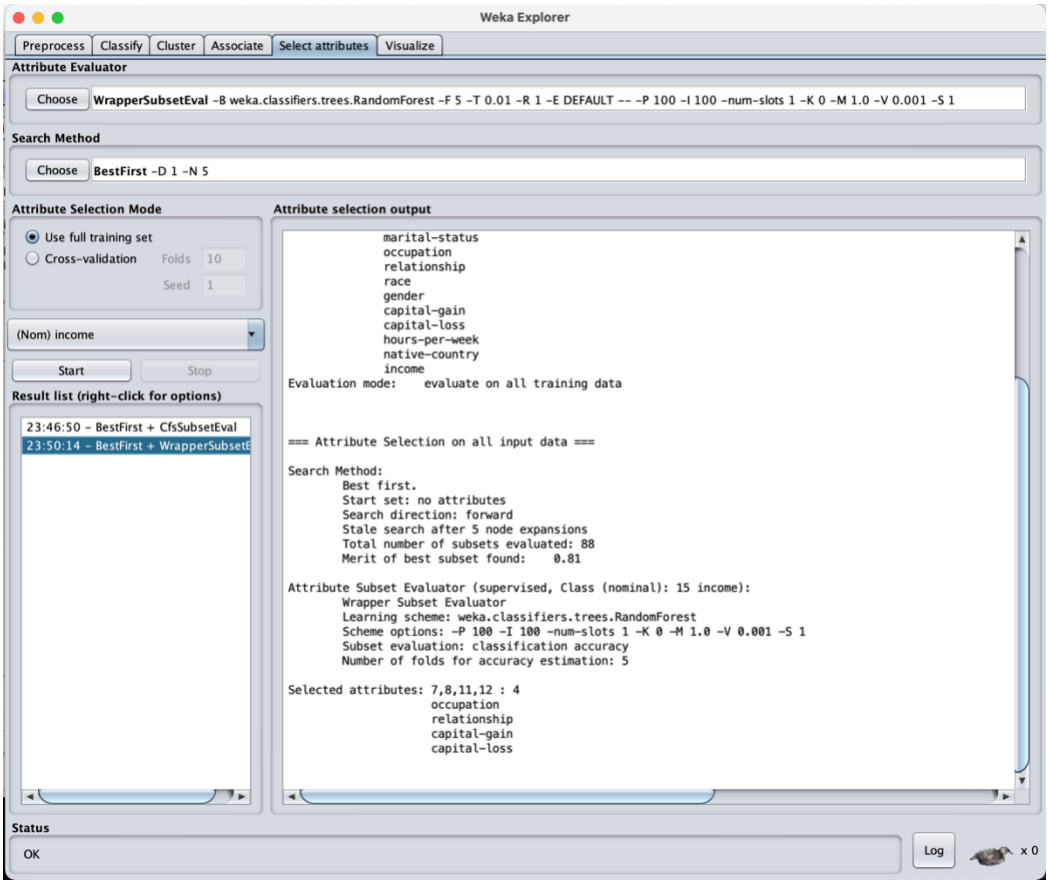
```
[29]: x1=df[['age','workclass','educational-num','capital-gain','capital-loss','hours-per-week']]
      y1=df[['income']]
      x1_train,x1_test,y1_train,y1_test=train_test_split(x1,y1,test_size=0.3,random_state=15)
      knn = KNeighborsClassifier()
      knn.fit(x1_train,y1_train.values.ravel())
      y1_test_predicted = knn.predict(x1_test)
      accuracy1 = metrics.accuracy_score(y1_test, y1_test_predicted)
      print("Before:",accuracy)
      print("After:",accuracy1)

      Before: 0.6230348598769652
      After: 0.7539302802460697
```

第二部份:**Weka**

載入資料集 **income.csv (**預測薪水是否能超過 **50K)**。

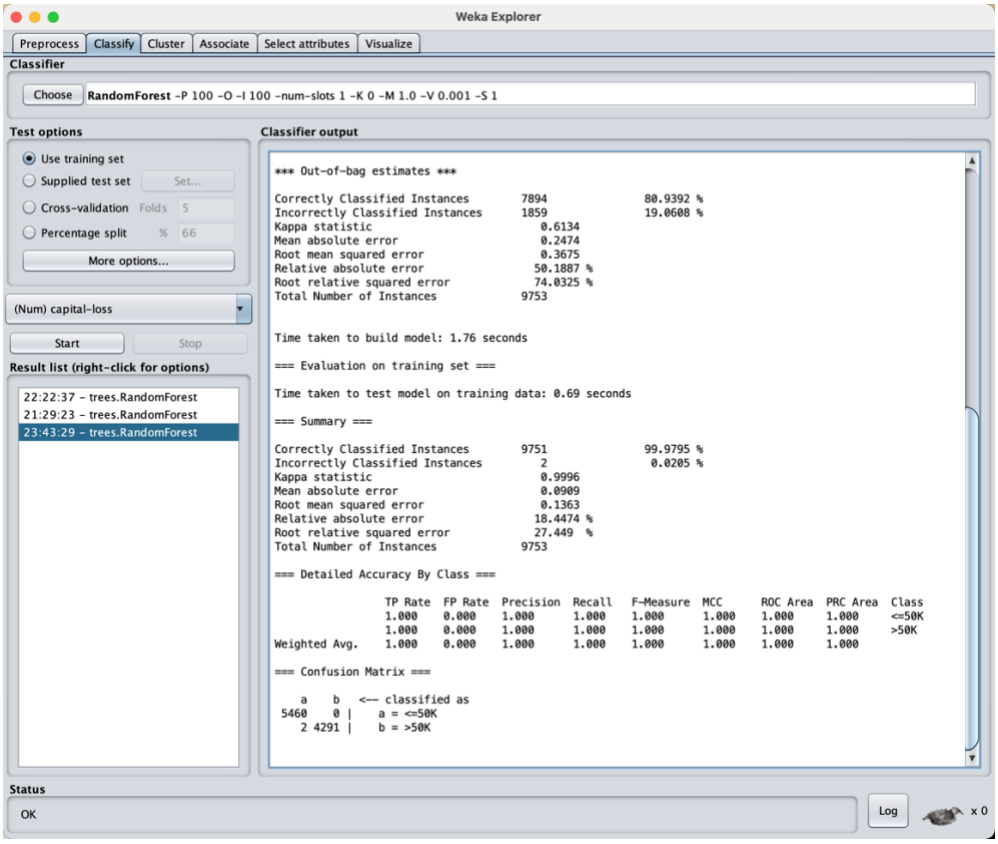
1. 針對 **RandomForest** 尋找最適合此模型的 **feature set**，並以 **OOB** 或 **5 CV**、**Accuracy** 為選擇的依據。請將過程與輸出結果截圖到作業中。**(20%)**



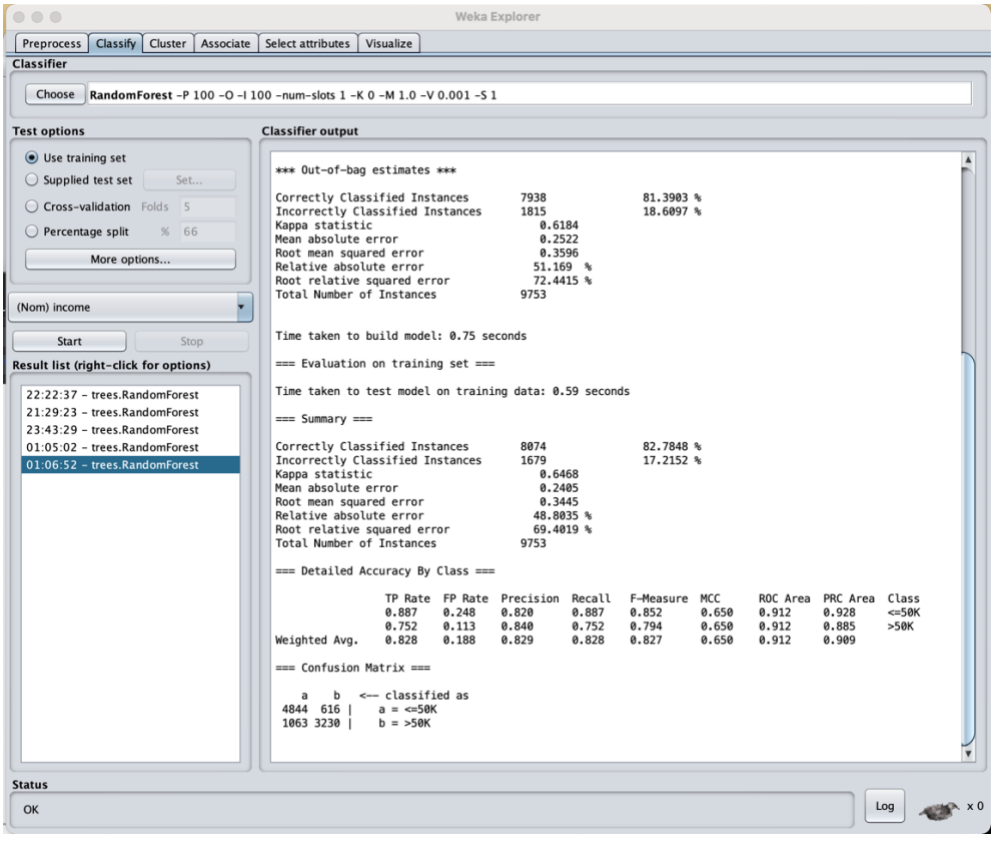
2. 請寫出挑選的 **feature set**。**(**並以 **weka** 輸出佐證**) (5%)**

   A: **4** 個（ occupation, relationship, capital-gain, capital-loss ）

3. 請使用挑選前與挑選後的 **features** 分別建立 **RandomForest (**使用 **OOB** 或 **5 CV)** 並做比較**(**至少寫出 **2** 點**)**，並將過程與輸出結果截圖到作業中。**(10%)**



Before



After

A:

1. 使用 feature set 之後整體的運算時間有顯著的提升
2. OOB 的 Correctly instance 也提升了 0.5%左右
3. Summary 的部分，由於我們在 After 只取用了 4 個 attribute 來建模型，所以在 Correctly instance 的部分則會有顯著的下降