|  |
| --- |
| 國立屏東大學資訊學院  資訊科學系 |
| Department of Computer Science,  National Pingtung University |

109學年度

機器學習期末報告

組員：戚凱源（CBE108024）

組員：張睿軒（CBE108023）

中 華 民 國 一零九 年 一 月

1. 資料集合一 (final\_project\_dataset\_1.csv)

一、資料分析與預處理

此資料集有6個自變量與1個應變量，自變量不進行刪去，全數套用。

*dataset=pd.read\_csv('final\_project\_dataset\_1.csv')*

*x=dataset.iloc[:,[0,1,2,3,4,5]].values*

*y=dataset.iloc[:,[6]].values*

其中，有3個自變量要打上虛擬標籤分別是'sex’ ‘smoker’ ‘country’

*from sklearn.preprocessing import LabelEncoder,OneHotEncoder*

*from sklearn.compose import ColumnTransformer*

*labelencoder\_x=LabelEncoder()*

*labelencoder\_x.fit\_transform(x[:,1])*

*x[:,1]=labelencoder\_x.fit\_transform(x[:,1])*

*ct=ColumnTransformer([("sex",OneHotEncoder(),[1])],remainder='passthrough')*

*X=ct.fit\_transform(x)*

#避免虛擬變量陷阱

*X=X[:, 1:]*

*labelencoder\_x1=LabelEncoder()*

*labelencoder\_x1.fit\_transform(x[:,4])*

*x[:,4]=labelencoder\_x1.fit\_transform(x[:,4])*

*ct1=ColumnTransformer([("smoker",OneHotEncoder(),[4])],remainder='passthrough')*

*X1=ct.fit\_transform(x)*

#避免虛擬變量陷阱

*X1=X1[:, 1:]*

*labelencoder\_x2=LabelEncoder()*

*labelencoder\_x2.fit\_transform(x[:,5])*

*x[:,5]=labelencoder\_x2.fit\_transform(x[:,5])*

*ct2=ColumnTransformer([("country",OneHotEncoder(),[5])],remainder='passthrough')*

*X2=ct.fit\_transform(x)*

#避免虛擬變量陷阱

*X2=X2[:, 1:]*

接著，分割測試集合與訓練集合其比例是80/20，不使用隨機狀態

*from sklearn.model\_selection import train\_test\_split*

*X\_train,X\_test,y\_train,y\_test=train\_test\_split(X2,y,test\_size=0.2,random\_state=0)*

因為資料變量之間量級差距，使用特徵縮放

*from sklearn.preprocessing import StandardScaler*

*sc\_x=StandardScaler()*

*X\_train=sc\_x.fit\_transform(X\_train)*

*X\_test=sc\_x.transform(X\_test)*

*labelencoder\_y=LabelEncoder()*

*y=labelencoder\_y.fit\_transform(y)*

至此，前處理完成

二、運用機器模型

因目的是預測可能罰金，本組採用簡單回歸、多項式回歸、反向淘汰、SVM與隨機森林

1.簡單回歸

*from sklearn.linear\_model import LinearRegression*

*regressor=LinearRegression()*

*regressor.fit(X\_train,y\_train*)

2.多項式回歸

本組試驗degree=2.3.4 其中4為最佳者

*from sklearn.linear\_model import LinearRegression*

*from sklearn.preprocessing import PolynomialFeatures*

*poly\_reg=PolynomialFeatures(degree=4)*

*x\_poly=poly\_reg.fit\_transform(X\_train)*

*lin\_reg2=LinearRegression()*

*lin\_reg2.fit(x\_poly,y\_train*)

3.反向淘汰

*import statsmodels.api as sm*

*X\_train=np.append(arr=np.ones((1070,1)).astype(int),values=X\_train,axis=1)*

#用反向淘汰建立模型

*X\_opt=X\_train[:,[0,1,2,3,4,5]]*

*X\_opt=np.array(X\_opt,dtype=float)*

*regressor\_OLS=sm.OLS(endog=y\_train,exog=X\_opt).fit()*

*regressor\_OLS.summary()*

*X\_opt=X\_train[:,[0,2,3,4,5]]*

*X\_opt=np.array(X\_opt,dtype=float)*

*regressor\_OLS=sm.OLS(endog=y\_train,exog=X\_opt).fit()*

*regressor\_OLS.summary()*

#利用反向淘汰留下的自變量跑模型

*X\_opt=X\_opt[:,[1]]*

*regressor.fit(X\_opt,y\_train)*

*X\_test=X\_test[:,[2]]*

*y\_pred\_pls=regressor.predict(X\_test*)

4.SVM

#在進行SVM時，程式希望進入測資的資料類別一樣，因此應變量的資料類別要更改後再切割測試與訓練集合

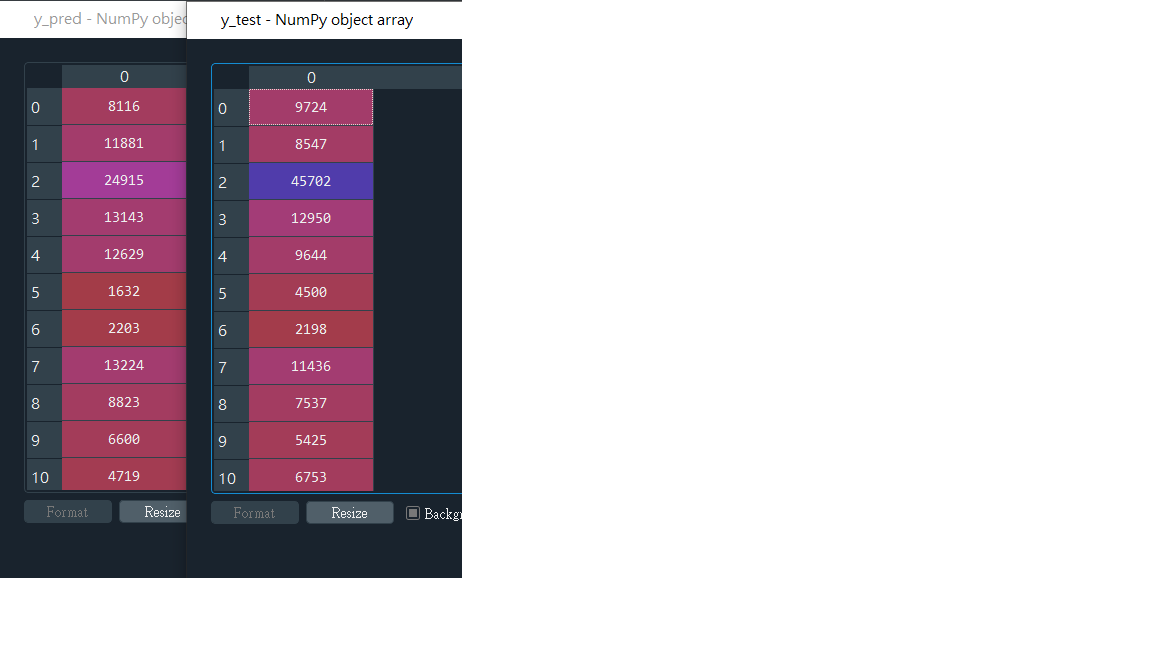
*y = np.array(y, dtype=int)*

*from sklearn.svm import SVC*

因為不是高複雜度資料kernel選擇'rbf’，隨機狀態為0

*classifier =SVC(kernel='rbf',random\_state=0)*

*classifier.fit(X\_train,y\_train)*



5.隨機森林

#相較於決策樹，隨機森林篩選出的分類器會更為精準

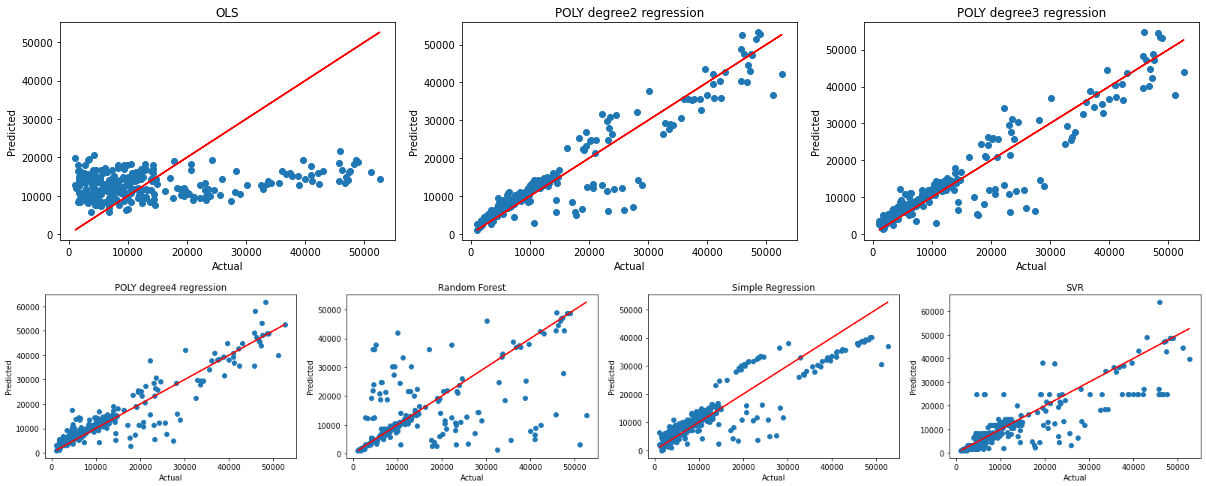
分類器數量為100，使用entropy衡量資料，隨機狀態為0

*from sklearn.ensemble import RandomForestClassifier*

*classifier=RandomForestClassifier(n\_estimators=100,criterion='entropy',random\_state=0)*

*classifier.fit(X\_train,y\_train*)

三、資料視覺化



因為是雜亂且不能回歸的資料，所有演算法都很難達到預測線，其中反向淘汰(OLS)在本來就不多的自變量中淘汰1個使其結果相差巨大而SVR、simple regression、在初期的預測度尚可。多項式(POLY)與隨機森林(RF)則達到較好的吻合度。輔以上方pred/test的數據，RF部分資料雖較為發散，卻十分吻合，相較之下，POLY有較多偏離數值，本組認為RF是最適合的演算法。

1. 資料集合二 (final\_project\_dataset\_2.csv)

一、資料分析與預處理

此資料有眾多自變量與1個應變量，以及龐大的資料與缺失資料，鑑於缺失資料極多，本組決定篩選自變量。

#日期不重要，地點與風向可以透過其他數值資料進行演算法歸納，而今天是否下雨與明天是否下雨邏輯上關聯度太低(應該要依靠溼度等數值)，最後我們只留下除了

”date” “location” "WindDir9am","WindDir3pm", "WindGustDir", "RainToday"

以外的自變量，同時把資料缺失過多的數據切除。

*dataset=pd.read\_csv('final\_project\_dataset\_2.csv')*

*dataset.drop(["Date","Location","WindDir9am","WindDir3pm", "WindGustDir", "RainToday"], axis = 1, inplace = True)*

*dataset.dropna(inplace=True)*

*dataset.RainTomorrow = [1 if each == "Yes" else 0 for each in dataset.RainTomorrow]*

*x= dataset.iloc[:,:-1].values#除了最後的所有DATA*

*y = dataset.iloc[:,[16]].values#最後的DATA*

接著，分割測試集合與訓練集合其比例是80/20，不使用隨機狀態

*from sklearn.model\_selection import train\_test\_split*

*X\_train,X\_test,y\_train,y\_test=train\_test\_split(X2,y,test\_size=0.2,random\_state=0)*

因為資料變量之間量級差距，使用特徵縮放

*from sklearn.preprocessing import StandardScaler*

*sc\_x=StandardScaler()*

*X\_train=sc\_x.fit\_transform(X\_train)*

*X\_test=sc\_x.transform(X\_test)*

至此，前處理完成

二、運用機器模型

目的在於預測是否明天會下雨，本組使用邏輯回歸、貝式定理、SVM與隨機森林。鑑於過多自變量，每種演算法都有使用主成分分析(PCA)，並對其使用優劣進行研究。

#已經由手肘定理得知component=2為最佳解

*from sklearn.decomposition import PCA*

*pca=PCA(n\_components=2)*

*X\_train=pca.fit\_transform(X\_train)*

*X\_test=pca.transform(X\_test)*

*explained\_variance=pca.explained\_variance\_ratio\_*

1.邏輯回歸

#隨機度為0

*from sklearn.linear\_model import LogisticRegression*

*classifier=LogisticRegression(random\_state=0)*

*classifier.fit(X\_train,y\_train)*

2.SVC

#使用linear模式建構模型，隨機度為0

*from sklearn.svm import SVC*

*classifier =SVC(kernel='linear',random\_state=0)*

*classifier.fit(X\_train,y\_train)*

3.貝式定理

*from sklearn.naive\_bayes import GaussianNB*

*classifier =GaussianNB()*

*classifier.fit(X\_train,y\_train)*

4.隨機森林

#相較於決策樹，隨機森林篩選出的分類器會更為精準

分類器數量為100，使用entropy衡量資料，隨機狀態為0

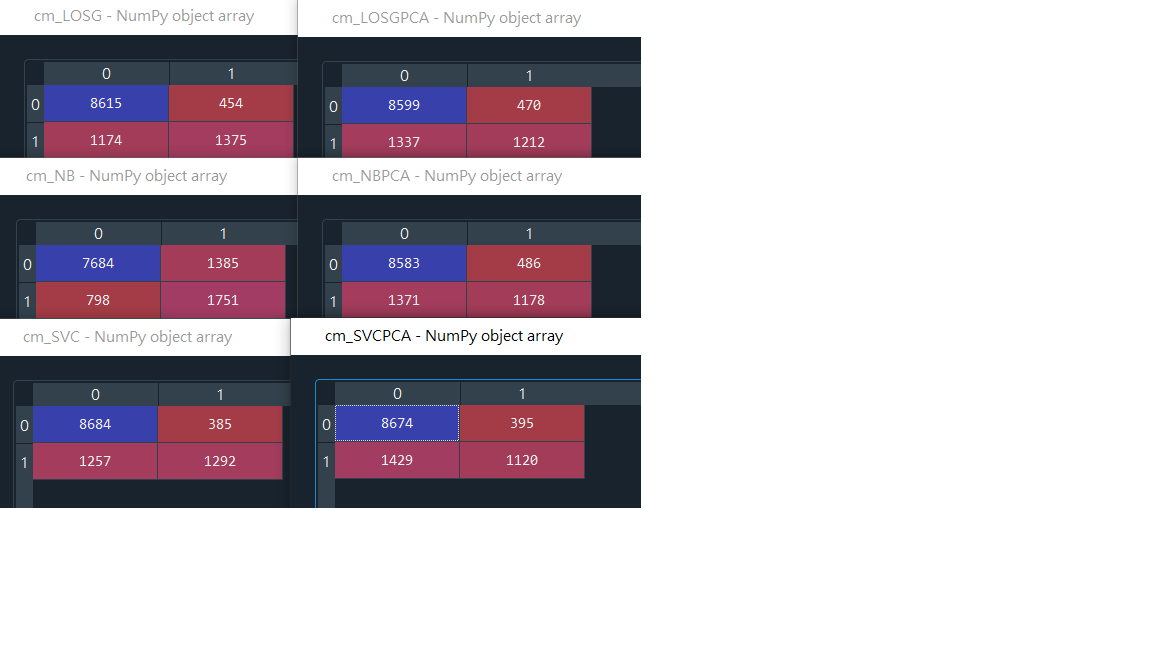
*from sklearn.ensemble import RandomForestClassifier*

*classifier=RandomForestClassifier(n\_estimators=100,criterion='entropy',random\_state=0)*

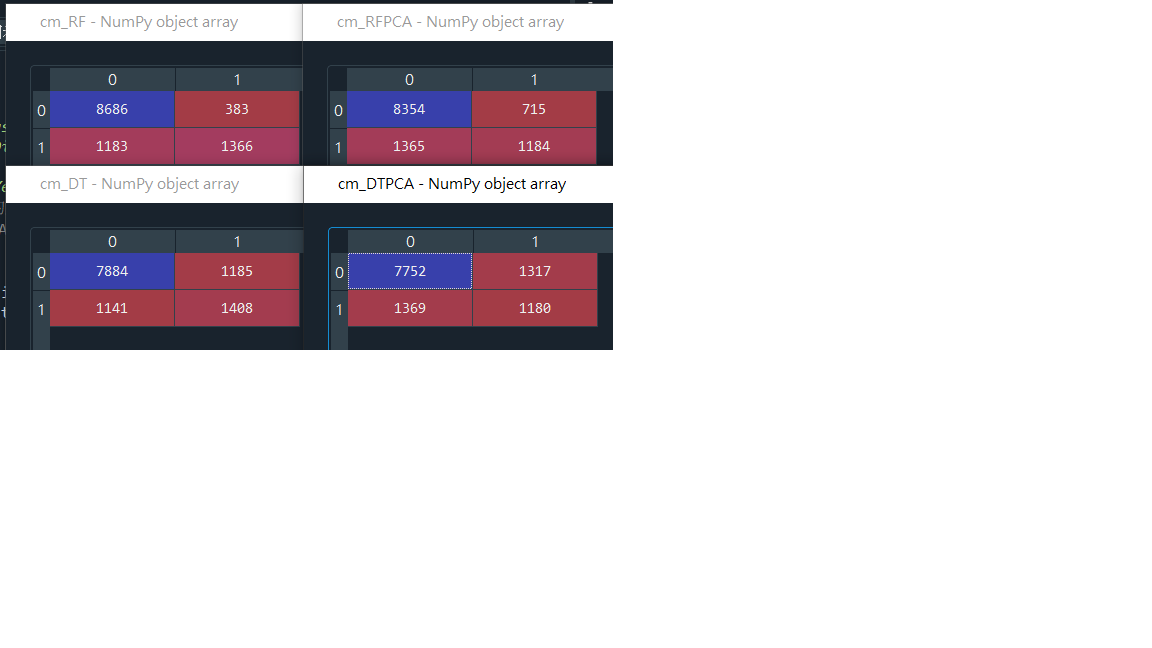
*classifier.fit(X\_train,y\_train)*

三、資料視覺化

用混淆矩陣(confusion matrix)把預測結果視覺化



就結果而言SVC、邏輯回歸(LOSG)、貝式(NB)的預測命中率各為86%、86%、81%其中貝式最差。



而隨機森林(RF)或是決策樹(DT)系列中命中率最高是RF，高達87%。就上述資料，使用RF可得較好結果

#這裡有一個小小迷思關於"要不要用RCA"， RF之前的PCA不是一個很好的優勢，PCA協助在OLS線性迴歸之前調整訓練特徵，這對於稀疏數據集非常需要。由於RF本身已經在沒有假設線性的情況下執行了良好/公平的正則化，所以它不一定是優勢。