Computation in Data Science 2021
Final Project

R10946013 劉謦瑄

1 Introduction

使用 GDSC cell line gene expression 來建立統計/機器學習模型，預測 25 個 patients 的藥物二元分類問題(resistant, sensitive)，並以 25 個 patients 的 labels 驗證模型的準確率，資料夾包含 2 個檔案 GDSC_PDX_Gemcitabine.csv 和 Patients_PDX_Gemcitabine.csv。

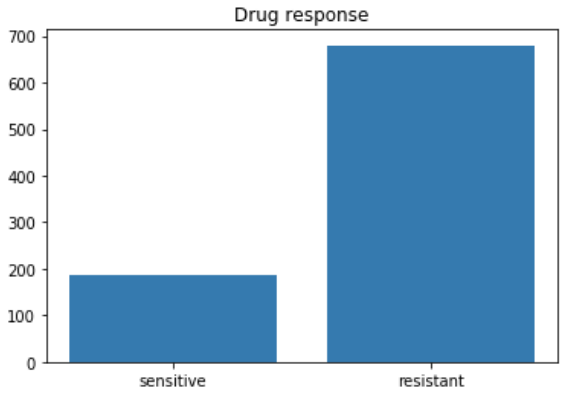
1.1 About Datasets

Training set：GDSC_PDX_Gemcitabine.csv 檔包含 866 個 GDSC cell lines gene expression levels 及 IC50。訓練資料表中共有 16192 個 features，PDX_Gemcitabine 的最大用藥濃度為為 1.024，若 IC50 小於最大用藥濃度，則標記為 S；若 IC50 大於最大用藥濃度，則標記為 R。
Testing set：Patients_PDX_Gemcitabine.csv 檔包含 25 個 patients' gene expression levels 及 labels。測試資料表中共有 16192 個 features。

然而，訓練資料表中的 features 數量雖然與測試資料表相同，但實際上卻並非都是相同的 features。因此在資料處理的過程中，我先篩選出了兩資料表中共同的 features（交集）來製作，一共是 16190 個 features。

1.2 Drug Response

Drug Response 的判斷方式是與 IC50 做比較，PDX_Gemcitabine 的最大用藥濃度為為 1.024，若 IC50 小於最大用藥濃度，則標記為 S（Sensitive）；若 IC50 大於最大用藥濃度，則標記為 R（Resistant）。訓練資料的判斷結果數量如下圖所示：



2 Data Processing

因為本次作業的資料表中提供很多種 features，對於模型的訓練成效不一定有好的幫助，因此我執行了 Feature Selection 的步驟。

為了比較多種 Feature Selection 方法對於模型訓練的成效，我透過 Python 的 `SelectFromModel` 與 `SelectKBest` 套件實作了以下四種特徵選擇的方法：

1. **Lasso 方法**：`SelectFromModel(LinearSVC(C=0.01, penalty="l1", dual=False).fit(X, y), prefit=True)`
2. **ANOVA 方法**：`SelectKBest(score_func=f_regression, k=378)`
3. **基於 Tree 的方法**：`SelectFromModel(ExtraTreesClassifier(n_estimators=50),prefit=True)`

下表是經過四種方法進行 Feature Selection 後的特徵數量結果：

Method	Feature 數量
Lasso	111
ANOVA	378
基於 Tree 的方法	5093

在 Feature Selection 後，我對數據資料進行**標準化（Standardization）**，再進行模型的訓練與資料預測。

```
# Standardization
scaler = StandardScaler()
df_train_lg = scaler.fit_transform(df_train_lg)
df_train_lasso = scaler.fit_transform(df_train_lasso)
df_train_ANOVA = scaler.fit_transform(df_train_ANOVA)
df_train_tree = scaler.fit_transform(df_train_tree)

print(df_train_lg.shape)
print(df_train_lasso.shape)
print(df_train_ANOVA.shape)
print(df_train_tree.shape)
```

```
(866, 6654)
(866, 111)
(866, 378)
(866, 5093)
```

接下來我會綜合考量訓練效能與模型訓練成效，針對訓練結果進行比較說明。

3 Model Training and Testing Result

在這個章節，我會討論有關於模型的訓練成果與 Test Set 預測的成效。
我使用了 **3 種**常見的 Machine Learning Classifier 來實作，透過 **Cross Validation** 的方式計算並分析模型訓練的 **Accuracy**，以及模型之 **Precision**、**Recall**、**F1-score** 等評估指標，並繪製**混淆矩陣（Confusion Matrix）**以及 **AUC** 指標與 **ROC 曲線**圖，以利評估模型訓練之成效。

另外，對於模型測試/預測的結果，我們也以上述方式進行呈現並分析比較。以下章節以 Classifier **分別比較各 Feature Selection 方法對於模型訓練的成效**。

● Hyper Parameter 調整

首先，針對各模型的主要參數進行調整與選取，我分別觀察了 RadomForest 與 XGBoost Classifier 中的 n_estimators 參數以及 K-Neighbors 模型中 n_neighbors 參數的大小與 accuracy 的變化，結果如下表。我將選取表現較優的該參數作為模型訓練時的參數設定值。

➢ RandomForest Classifier: n_estimators

n_estimators	10	20	30	40	50	60	70	80	90	100
accuracy	0.7690	0.7875	0.7840	0.7875	0.7886	0.7909	0.7933	**0.7956**	0.7921	0.7956

➢ XGBoost Classifier : n_estimators

n_estimators	10	20	30	40	50	60	70	80	90	100
accuracy	0.7910	0.7979	0.7990	0.7921	0.7910	0.7956	0.7953	0.7967	**0.801**	0.7991

➢ K-Neighbors Classifier : n_neighbors

n_neighbors	10	20	30	40	50	60	70	80	90	100
accuracy	0.7875	**0.8009**	0.7875	0.7852	0.7852	0.7852	0.7852	0.7852	0.7852	0.7852

3.1 RandomForest Classifier

- Feature Selection – **LASSO**

  ✓ Cross Validation (5-fold) (for-loop 5 times):

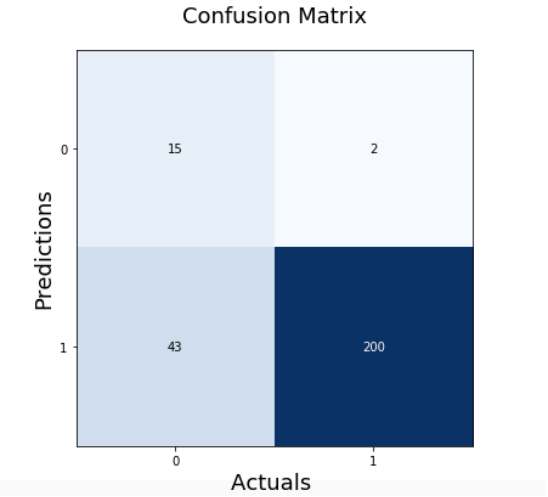
```
[0.77586207 0.78612717 0.79768786 0.78034682 0.80346821]
[0.79310345 0.79190751 0.78034682 0.77456647 0.79768786]
[0.75862069 0.78034682 0.79190751 0.78034682 0.8150289 ]
[0.75862069 0.77456647 0.79190751 0.78612717 0.80924855]
[0.7816092  0.78034682 0.77456647 0.79768786 0.78612717]
```

  → Average Score of Cross Validation: `0.7859265165105308`

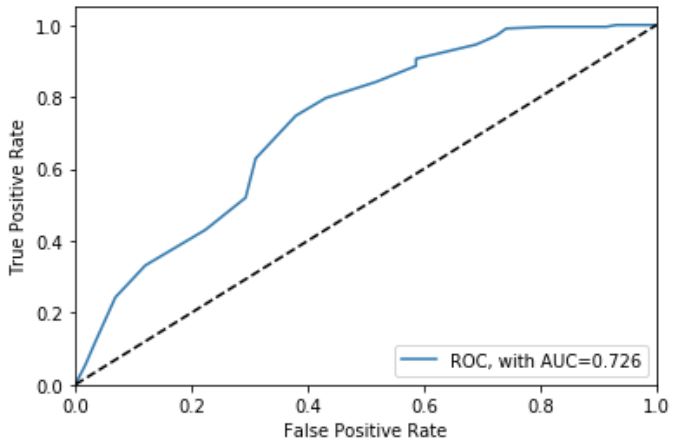
  ✓ Accuracy Score: `0.8269230769230769`

  ✓ Classification Report and Confusion Matrix:

	precision	recall	f1-score	support
0	0.88	0.26	0.40	58
1	0.82	0.99	0.90	202
accuracy			0.83	260
macro avg	0.85	0.62	0.65	260
weighted avg	0.84	0.83	0.79	260



  ✓ AUC & ROC Curve: → AUC Score: 0.726



- Feature Selection – **ANOVA**

  ✓ Cross Validation (5-fold) (for-loop 5 times):

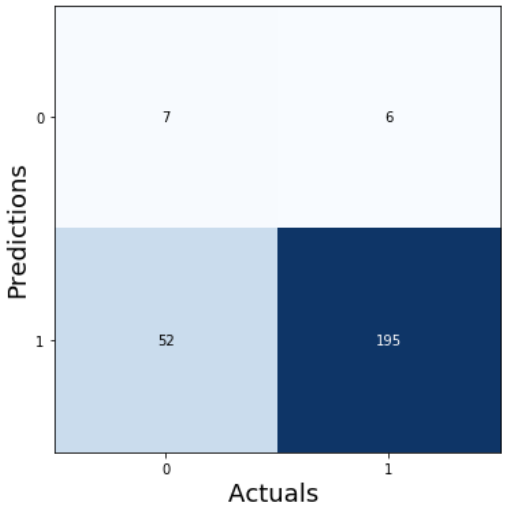
```
[0.7816092  0.79768786 0.78034682 0.78034682 0.79768786]
[0.7816092  0.79768786 0.76878613 0.78612717 0.79768786]
[0.75862069 0.78612717 0.78034682 0.78034682 0.79190751]
[0.78735632 0.78612717 0.78034682 0.78034682 0.80346821]
[0.79310345 0.79190751 0.79190751 0.78034682 0.79768786]
```

  → Average Score of Cross Validation: `0.786380971364029`

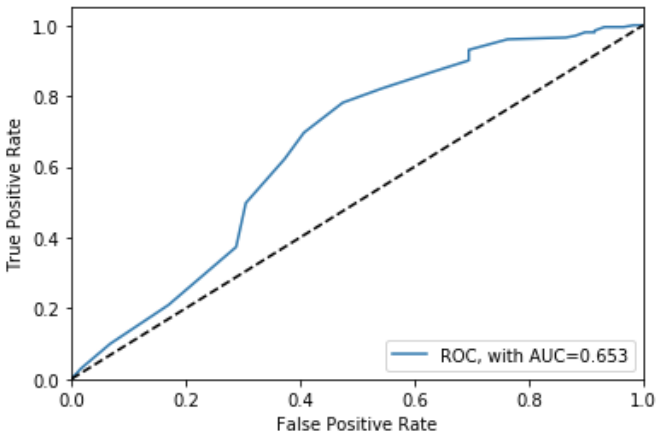
  ✓ Accuracy Score: `0.7769230769230769`

  ✓ Classification Report and Confusion Matrix:

	precision	recall	f1-score	support
0	0.54	0.12	0.19	59
1	0.79	0.97	0.87	201
accuracy			0.78	260
macro avg	0.66	0.54	0.53	260
weighted avg	0.73	0.78	0.72	260



  ✓ AUC & ROC Curve: → AUC Score: 0.653



- Feature Selection – **基於 Tree 的方法**

  ✓ Cross Validation (5-fold) (for-loop 5 times):

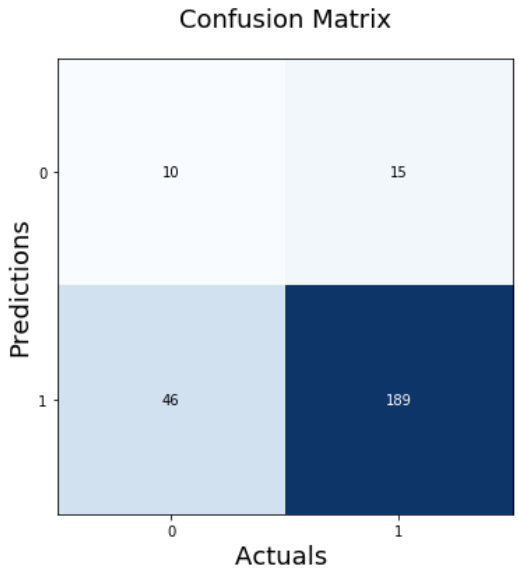
```
[0.72988506 0.79190751 0.76878613 0.74566474 0.76878613]
[0.75862069 0.78034682 0.77456647 0.77456647 0.76300578]
[0.6954023  0.76300578 0.75144509 0.78034682 0.78034682]
[0.76436782 0.78612717 0.75144509 0.78612717 0.76300578]
[0.74712644 0.73988439 0.77456647 0.75722543 0.74566474]
```

  → Average Score of Cross Validation: `0.7616889243239651`

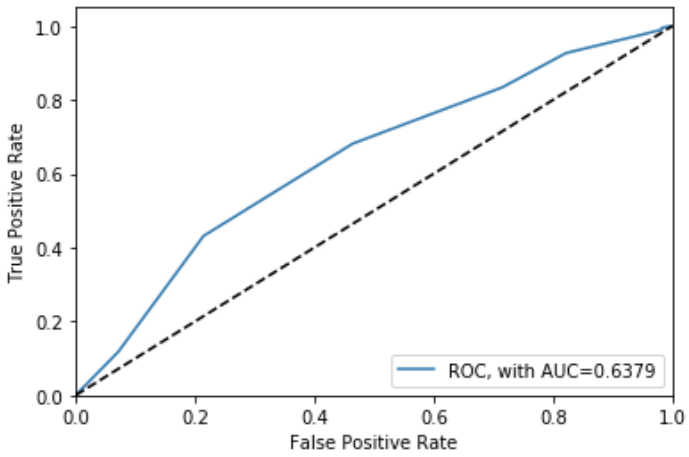
  ✓ Accuracy Score: `0.7653846153846153`

  ✓ Classification Report and Confusion Matrix:

	precision	recall	f1-score	support
0	0.40	0.18	0.25	56
1	0.80	0.93	0.86	204
accuracy			0.77	260
macro avg	0.60	0.55	0.55	260
weighted avg	0.72	0.77	0.73	260



  ✓ AUC & ROC Curve: → AUC Score: 0.6379



3.2 XGBoost Classifier

- Feature Selection – **LASSO**

  ✓ Cross Validation (5-fold) (for-loop 5 times):

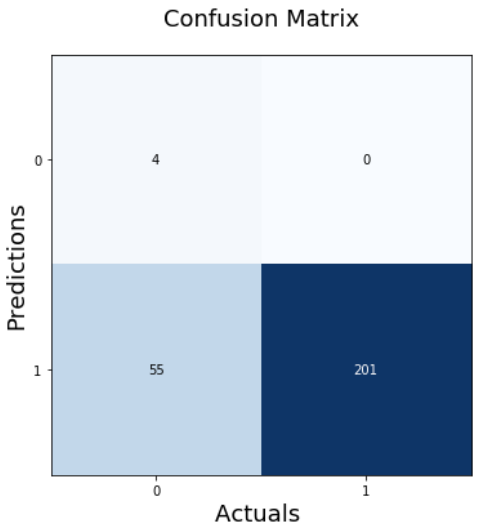
```
[0.75862069 0.80924855 0.77456647 0.79768786 0.8150289 ]
[0.75862069 0.80924855 0.77456647 0.79768786 0.8150289 ]
[0.75862069 0.80924855 0.77456647 0.79768786 0.8150289 ]
[0.75862069 0.80924855 0.77456647 0.79768786 0.8150289 ]
[0.75862069 0.80924855 0.77456647 0.79768786 0.8150289 ]
```

  → Average Score of Cross Validation: `0.7910304963125374`

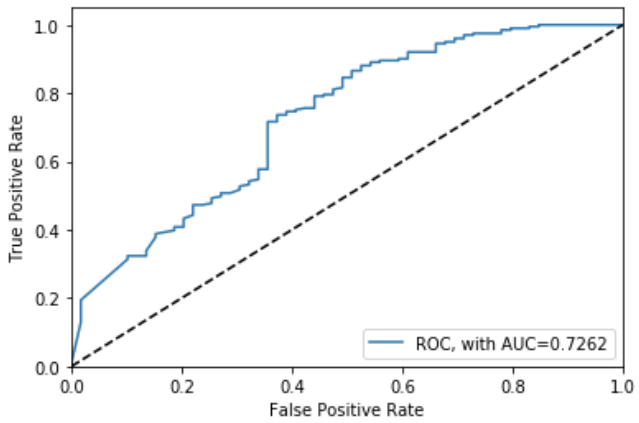
  ✓ Accuracy Score: `0.7884615384615384`

  ✓ Classification Report and Confusion Matrix:

	precision	recall	f1-score	support
0	1.00	0.07	0.13	59
1	0.79	1.00	0.88	201
accuracy			0.79	260
macro avg	0.89	0.53	0.50	260
weighted avg	0.83	0.79	0.71	260



  ✓ AUC & ROC Curve: → AUC Score: 0.726



- Feature Selection – **ANOVA**

  ✓ Cross Validation (5-fold) (for-loop 5 times):

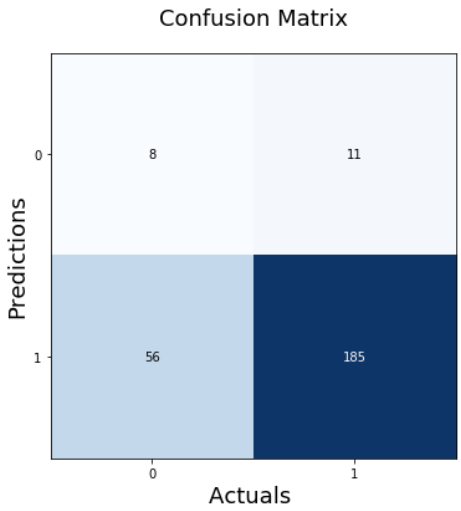
```
[0.75862069 0.80924855 0.77456647 0.79768786 0.8150289 ]
[0.75862069 0.80924855 0.77456647 0.79768786 0.8150289 ]
[0.75862069 0.80924855 0.77456647 0.79768786 0.8150289 ]
[0.75862069 0.80924855 0.77456647 0.79768786 0.8150289 ]
[0.75862069 0.80924855 0.77456647 0.79768786 0.8150289 ]
```

  → Average Score of Cross Validation: `0.7910304963125374`

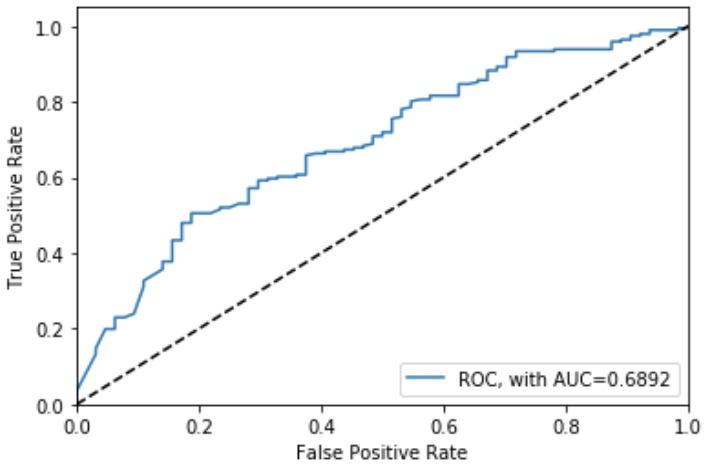
  ✓ Accuracy Score: `0.7423076923076923`

  ✓ Classification Report and Confusion Matrix:

	precision	recall	f1-score	support
0	0.42	0.12	0.19	64
1	0.77	0.94	0.85	196
accuracy			0.74	260
macro avg	0.59	0.53	0.52	260
weighted avg	0.68	0.74	0.69	260



  ✓ AUC & ROC Curve: → AUC Score: 0.689



- Feature Selection – **基於 Tree 的方法**

  ✓ Cross Validation (5-fold) (for-loop 5 times):

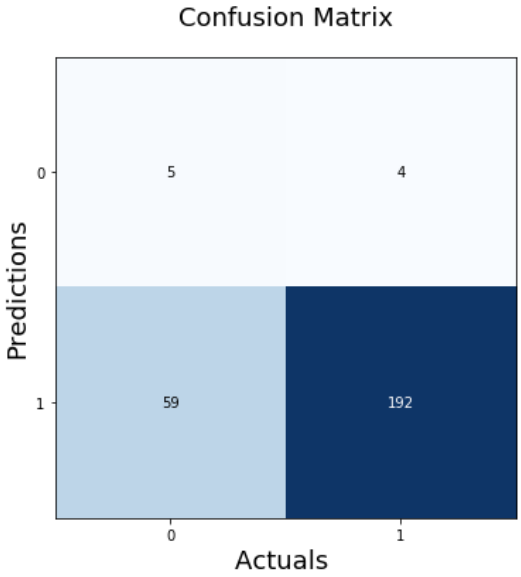
```
[0.75862069 0.80924855 0.77456647 0.79768786 0.8150289 ]
[0.75862069 0.80924855 0.77456647 0.79768786 0.8150289 ]
[0.75862069 0.80924855 0.77456647 0.79768786 0.8150289 ]
[0.75862069 0.80924855 0.77456647 0.79768786 0.8150289 ]
[0.75862069 0.80924855 0.77456647 0.79768786 0.8150289 ]
```

  → Average Score of Cross Validation: `0.7910304963125374`

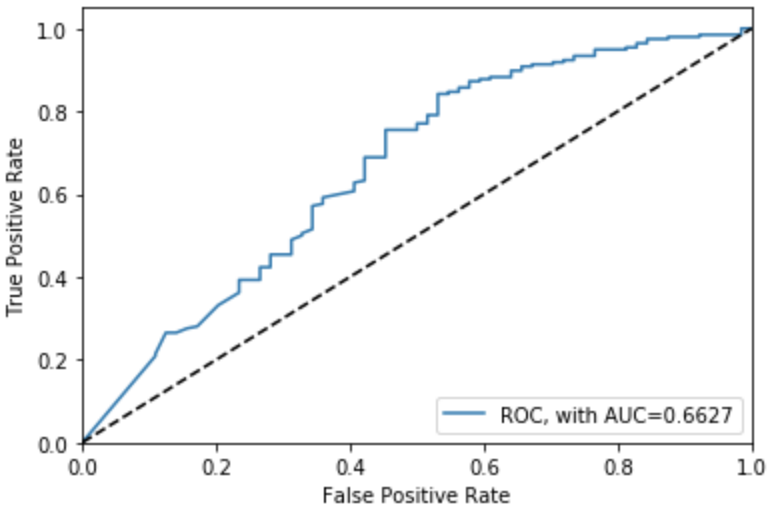
  ✓ Accuracy Score: `0.7576923076923077`

  ✓ Classification Report and Confusion Matrix:

	precision	recall	f1-score	support
0	0.56	0.08	0.14	64
1	0.76	0.98	0.86	196
accuracy			0.76	260
macro avg	0.66	0.53	0.50	260
weighted avg	0.71	0.76	0.68	260



  ✓ AUC & ROC Curve: → AUC Score: 0.6627



3.3 K-Neighbors Classifier

- Feature Selection – **LASSO**

  ✓ Cross Validation (5-fold) (for-loop 5 times):

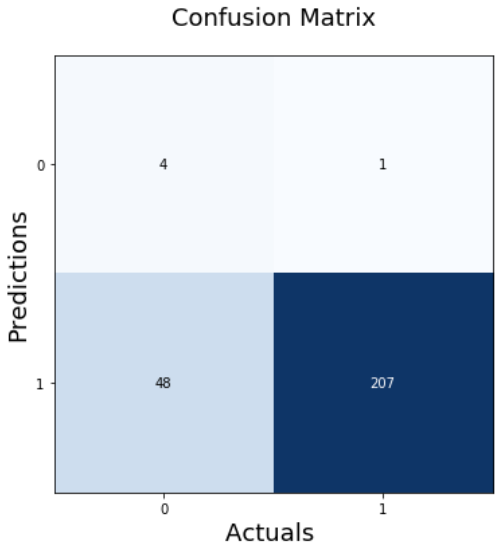
```
[0.78735632 0.79190751 0.76878613 0.78034682 0.80924855]
[0.78735632 0.79190751 0.76878613 0.78034682 0.80924855]
[0.78735632 0.79190751 0.76878613 0.78034682 0.80924855]
[0.78735632 0.79190751 0.76878613 0.78034682 0.80924855]
[0.78735632 0.79190751 0.76878613 0.78034682 0.80924855]
```

  → Average Score of Cross Validation: `0.7875290678360243`

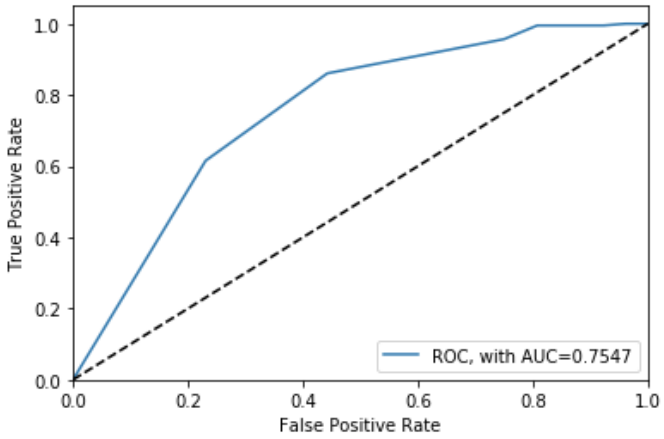
  ✓ Accuracy Score: `0.8115384615384615`

  ✓ Classification Report and Confusion Matrix:

	precision	recall	f1-score	support
0	0.80	0.08	0.14	52
1	0.81	1.00	0.89	208
accuracy			0.81	260
macro avg	0.81	0.54	0.52	260
weighted avg	0.81	0.81	0.74	260



  ✓ AUC & ROC Curve: → AUC Score: 0.7547



- Feature Selection – **ANOVA**

  ✓ Cross Validation (5-fold) (for-loop 5 times):

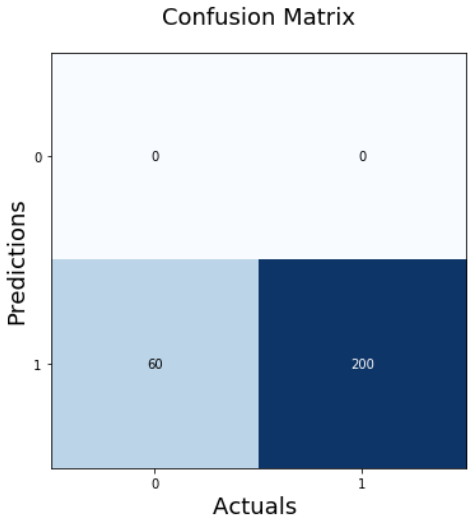
```
[0.7816092  0.78612717 0.78612717 0.78612717 0.78612717]
[0.7816092  0.78612717 0.78612717 0.78612717 0.78612717]
[0.7816092  0.78612717 0.78612717 0.78612717 0.78612717]
[0.7816092  0.78612717 0.78612717 0.78612717 0.78612717]
[0.7816092  0.78612717 0.78612717 0.78612717 0.78612717]
```

  → Average Score of Cross Validation: `0.7852235731845061`

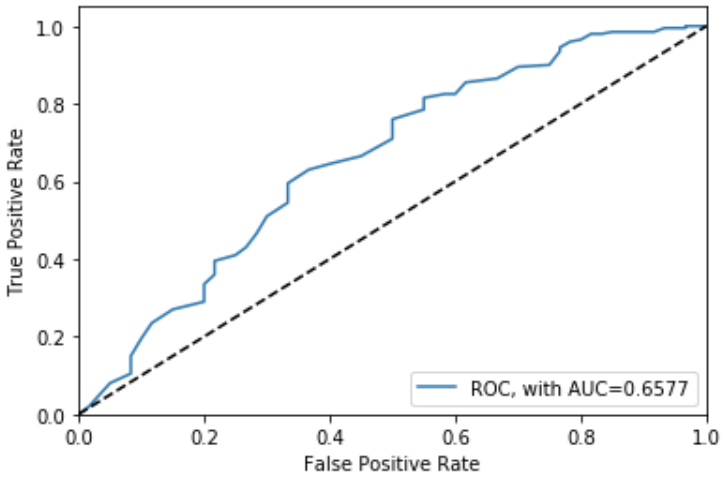
  ✓ Accuracy Score: `0.7692307692307693`

  ✓ Classification Report and Confusion Matrix:

	precision	recall	f1-score	support
0	0.00	0.00	0.00	60
1	0.77	1.00	0.87	200
accuracy			0.77	260
macro avg	0.38	0.50	0.43	260
weighted avg	0.59	0.77	0.67	260



  ✓ AUC & ROC Curve: → AUC Score: 0.6577



- Feature Selection – **基於 Tree 的方法**

  ✓ Cross Validation (5-fold) (for-loop 5 times):

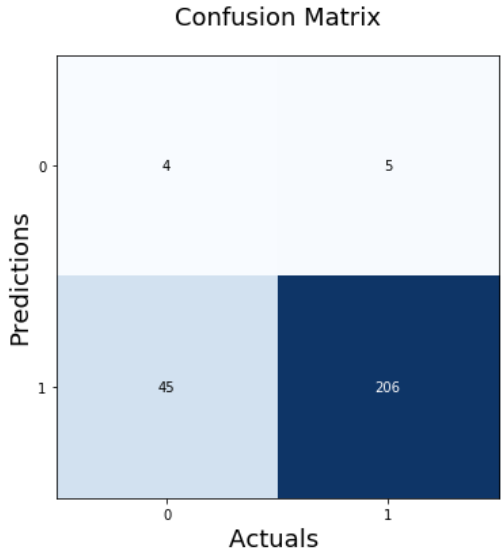
```
[0.78735632 0.79190751 0.76878613 0.78034682 0.80924855]
[0.78735632 0.79190751 0.76878613 0.78034682 0.80924855]
[0.78735632 0.79190751 0.76878613 0.78034682 0.80924855]
[0.78735632 0.79190751 0.76878613 0.78034682 0.80924855]
[0.78735632 0.79190751 0.76878613 0.78034682 0.80924855]
```

  → Average Score of Cross Validation: `0.7875290678360243`

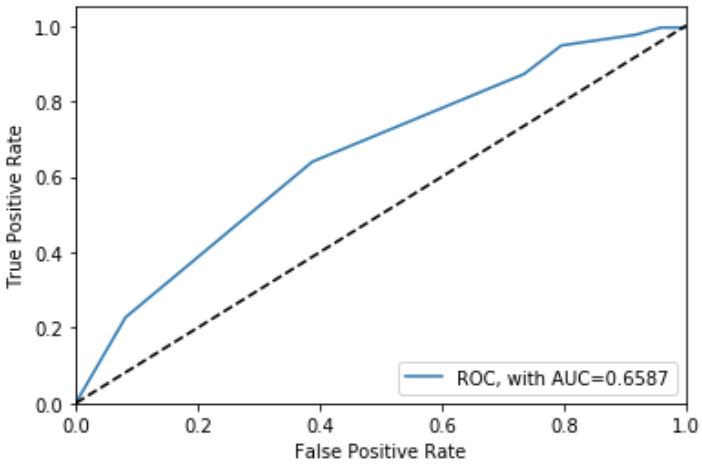
  ✓ Accuracy Score: `0.8076923076923077`

  ✓ Classification Report and Confusion Matrix:

	precision	recall	f1-score	support
0	0.44	0.08	0.14	49
1	0.82	0.98	0.89	211
accuracy			0.81	260
macro avg	0.63	0.53	0.51	260
weighted avg	0.75	0.81	0.75	260



  ✓ AUC & ROC Curve: → AUC Score: 0.6587



3.4 Results and Comparison

- **Accuracy Score**

	LASSO		ANOVA		TREE	
	Training	**Testing**	**Training**	**Testing**	**Training**	**Testing**
Ramdom Forest	**0.8269**	0.5	**0.7269**	0.48	**0.7654**	0.5238
XGBoost	**0.7885**	0.5278	**0.7423**	0.45	**0.7577**	0.5
K-Neighbors	**0.8115**	0.49	**0.7692**	0.56	**0.8077**	0.5

上表是針對各 Model 在不同 Feature Selection 方法下的 Training 和 Testing Accuracy Score 的比較。下一章節我會針對模型結果進行說明。

4 Conclusion and Discussion

本次作業我使用了**3種**常見的Machine Learning Classifier來實作，透過多次**Cross Validation**的方式計算並分析模型訓練的**Accuracy**，以及模型之**Precision**、**Recall**、**F1-score**等評估指標，並繪製**混淆矩陣（Confusion Matrix）**以及**AUC**指標與**ROC曲線**圖，比較各模型在不同Feature Selection下的成效。

我選擇了以下三種不同的Feature Selection方法來進行模型訓練：

Method	Feature 數量
Lasso	111
ANOVA	378
基於Tree的方法	5093

表中是各種Feature Selection後的特徵數量，數量大小為：LASSO<ANOVA<Tree。

而從模型訓練的成效，我們可以看到，雖然LASSO的Feature數量最少（僅有111個），但在模型訓練的成效上卻是表現相對亮眼的，訓練速度更是這三種之中最快的，這也真實地反應了LASSO的原理。不管是哪一種分類模型，我們都可以看到在Feature Selection方法中，LASSO方法的表現都是最佳的。

另外，綜觀所有的訓練結果分數，我們也可以看到在各種Feature Selection的方法下，通常Random Forest與K-Neighbors的表現都相對較佳，Testing Accuracy Score也都與其他模型結果相差不多。

然而綜合模型效能考量，LASSO能使用最少數量的特徵，達到與其他模型差不多的成效。在最小的模型訓練時間與成本的優勢下，本次實驗會推薦「選用LASSO方法進行Feature Selection，並使用Random Forest Classifier進行分類預測」的方法。