

# 二元分類教學 ( CLF101 ) - 初級

 Open in Colab

([https://colab.research.google.com/github/y-s-liu/machine-learning-tutorial/blob/master/PyCaretT\\_Binary\\_Classification\\_Tutorial\\_Level\\_Beginner\\_CLF101.ipynb](https://colab.research.google.com/github/y-s-liu/machine-learning-tutorial/blob/master/PyCaretT_Binary_Classification_Tutorial_Level_Beginner_CLF101.ipynb)).

使用版本: PyCaret 2.2.1

參考來源: [Binary Classification Tutorial Level Beginner - CLF101.ipynb](#)

(<https://github.com/pycaret/pycaret/blob/master/tutorials/Binary%20Classification%20Tutorial%20Level%20%20CLF101.ipynb>)

## 1.0 學習目標

本課程使用 "pycaret.classification" 模組來進行二元分類。

在這個課程將會學到的項目如下

### A. 初始化

- 獲取分析資料：使用pandas函式庫讀取檔案，也可從PyCaret資料集導入資料。
- 設置實驗環境：在PyCaret中設置實驗用以開始構建分類模型。

### B. 訓練模型

- 建構模型：建構模型，執行分層交叉驗證和評估分類指標。
- 調整模型：自動調整分類模型的超參數。

### C. 模型分析

- 分析模型性能：使用各種圖形分析模型性能。

### D. 模型運用

- 部署模型確定：在實驗結束時最終確定最佳模型。
- 模型預測：對未知的資料進行預測。
- 模型保存/載入：保存/載入模型以備將來使用。

## 1.1 安裝 PyCaret 函式庫

```
!pip install pycaret
```

## 1.2 Pre-Requisites

- Python 3.6 或更新版本
- PyCaret 2.2.1 或更新版本
- 二元分類的基礎知識

## 1.3 For Google colab users:

若是要在 Google colab 執行，可使用下列程式碼開啟 notebook 圖形互動功能。

```
from pycaret.utils import enable_colab  
enable_colab()
```

## 2.0 什麼是二元分類？

二元分類是一種監督式機器學習技術，其目的是預測分類的類別標籤，例如通過/失敗，正/負，缺陷/非缺陷等。

下面列出了一些實際的分類範例：

- 確定患者是否患有某種疾病的醫學測試 - 疾病是否存在。
- 工廠中產線的品質控制，即確定產品是否滿足規格 - 通過/不通過。

## 3.0 PyCaret中分類模組的概述

PyCaret的分類模組（`pycaret.classification`）是一種監督式機器學習的模組，可使用各種技術和演算法將資料分類為二進位的結果。它具有18種演算法和15種圖形來分析模型的性能。常見用法包括預測客戶違約（是或否），客戶流失（客戶將離開或留下），發現的疾病（陽性或陰性）。

## 4.0 資料集說明

此課程，將使用來自UCI的名為“信用卡客戶資料集”的資料集，此資料可看出台灣客戶的違約支付情況。該資料集包含有關2005年4月至2005年9月台灣地區信用卡客戶的信用額度、個人資料、付款歷史記錄和帳單等資訊。有24,000筆樣本和25筆特徵。

每列的簡短描述如下：

- **ID:** 每個客戶的識別碼
- **LIMIT\_BAL:** 新台幣的信用額度（包括個人和家庭/補充信用）
- **SEX:** 性別（1 =男性，2 =女性）
- **EDUCATION:** 教育程度（1=碩士，2=大學，3=高中，4=其他，5=未知，6=未知）
- **MARRIAGE:** 婚姻狀況（1 =已婚，2 =單身，3 =其他）
- **AGE:** 年齡
- **PAY\_0 to PAY\_6:** n個月前的還款狀態（PAY\_0=上個月 .. PAY\_6 = 6個月前）（標籤：-1=正常付款，1=延遲一個月付款，2=延遲兩個月付款 .. 8=延遲八個月付款，9=延遲九個月及以上的付款）
- **BILL\_AMT1 to BILL\_AMT6:** n個月前的帳單金額（BILL\_AMT1=上個月 .. BILL\_AMT6=6個月前）
- **PAY\_AMT1 to PAY\_AMT6:** n個月前的付款金額（BILL\_AMT1=上個月 .. BILL\_AMT6=6個月前）
- **default :** 正常付款（1 =是，0 =否）“目標列(target column)”

資料集來源：

Lichman, M. (2013). UCI Machine Learning Repository. Irvine, CA: University of California, School of Information and Computer Science.

資料來源 (<https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients>)

## 5.0 獲取資料

可以從[此處 \(https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients\)](https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients)找到的原始下載資料，然後使用pandas進行載入（了解操作方法），也可以使用PyCaret的資料集儲存庫直接透過 `get_data` 函式獲取資料（這將需要網際網路連接）。

In [3]:

```
from pycaret.datasets import get_data
dataset = get_data('credit')
```

	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_1	PAY_2	PAY_3	PAY_4	PAY_5
0	20000	2	2	1	24	2	2	-1	-1	-2
1	90000	2	2	2	34	0	0	0	0	0
2	50000	2	2	1	37	0	0	0	0	0
3	50000	1	2	1	57	-1	0	-1	0	0
4	50000	1	1	2	37	0	0	0	0	0

◀ ▶

In [4]:

```
#check the shape of data
dataset.shape
```

Out[4]:

(24000, 24)

為了用於 predict\_model 函式的測試資料集，已從原始資料集中保留了1200筆資料的樣本以用於預測。不應用訓練/測試資料集混淆，執行資料集分割是為了模擬現實生活場景。因此，在進行機器學習實驗時，不應用到1200筆資料。

In [5]:

```
data = dataset.sample(frac=0.95, random_state=786)
data_unseen = dataset.drop(data.index)
data.reset_index(inplace=True, drop=True)
data_unseen.reset_index(inplace=True, drop=True)
print('Data for Modeling: ' + str(data.shape))
print('Unseen Data For Predictions: ' + str(data_unseen.shape))
```

Data for Modeling: (22800, 24)  
 Unseen Data For Predictions: (1200, 24)

## 6.0 設定 PyCaret 環境

在pycaret中，使用 setup 函式初始化環境並建立轉換管道，用以準備用於建立模型和模型部署的資料。執行任何其他功能之前，必須先使用 setup 。它有兩個必填參數：pandas 框架的資料(dataframe)以及分類標籤(target)的名稱。其他參數都是選項，用於自行定義前處理管道。

執行 setup 函式時，PyCaret的推理演算法將根據某些屬性，自動推斷所有功能的資料類型。然而，理論上來說應該正確推斷資料類型，但並非總是如此。為了解決這個問題，在執行 setup 函式之後，PyCaret將顯示一個包含功能及其推斷資料類型的表。如果正確識別了所有資料類型，則可以按Enter鍵繼續，反之則鍵入quit結束實驗。對於任何機器學習實驗，確保資料類型正確，方能做出正確的實現，確認PyCaret自動執行一些前處理任務的過程是否正確，此步驟必不可少。

In [6]:

```
from pycaret.classification import *
```

In [7]:

```
exp_clf101 = setup(data = data, target = 'default', session_id=123)
```

	Description	Value
0	session_id	123
1	Target	default
2	Target Type	Binary
3	Label Encoded	0: 0, 1: 1
4	Original Data	(22800, 24)
5	Missing Values	False
6	Numeric Features	14
7	Categorical Features	9
8	Ordinal Features	False
9	High Cardinality Features	False
10	High Cardinality Method	None
11	Transformed Train Set	(15959, 88)
12	Transformed Test Set	(6841, 88)
13	Shuffle Train-Test	True
14	Stratify Train-Test	False
15	Fold Generator	StratifiedKFold
16	Fold Number	10
17	CPU Jobs	-1
18	Use GPU	False
19	Log Experiment	False
20	Experiment Name	clf-default-name
21	USI	e9de
22	Imputation Type	simple
23	Iterative Imputation Iteration	None
24	Numeric Imputer	mean
25	Iterative Imputation Numeric Model	None
26	Categorical Imputer	constant
27	Iterative Imputation Categorical Model	None
28	Unknown Categoricals Handling	least_frequent
29	Normalize	False
30	Normalize Method	None
31	Transformation	False
32	Transformation Method	None
33	PCA	False
34	PCA Method	None
35	PCA Components	None
36	Ignore Low Variance	False
37	Combine Rare Levels	False

	Description	Value
38	Rare Level Threshold	None
39	Numeric Binning	False
40	Remove Outliers	False
41	Outliers Threshold	None
42	Remove Multicollinearity	False
43	Multicollinearity Threshold	None
44	Clustering	False
45	Clustering Iteration	None
46	Polynomial Features	False
47	Polynomial Degree	None
48	Trigonometry Features	False
49	Polynomial Threshold	None
50	Group Features	False
51	Feature Selection	False
52	Features Selection Threshold	None
53	Feature Interaction	False
54	Feature Ratio	False
55	Interaction Threshold	None
56	Fix Imbalance	False
57	Fix Imbalance Method	SMOTE

成功執行 `setup` 函式後，它將顯示一些重要資訊的網格。大多數資訊與執行 `setup` 函式時建構的前處理管道有關。這些功能的使用，大多數功能超出了此次教學的範圍，但是為現階段要注意的一些重要事項包括以下幾點：

- **session\_id**：一個虛擬隨機數作為隨機變數的種子分佈在所有函數中，方便以後可以重現實驗。如果沒有設定此變數“`session_id`”，則會自動生成一個隨機數，該隨機數將分配給所有函數。在此實驗中，將“`session_id`”設置為“123”。
- **Target Type**：二元或多類分類。其目標類型將被自動檢測並顯示。對於實驗的執行方式沒有區別。所有功能都是相同的。
- **Label Encoded**：當Target變量的類型為字元（即“是”或“否”）而不是1或0時，它將自動將標籤編碼為1和0，並顯示映射（0：否，1：是）以供參考。在此實驗中，由於目標變量是數字類型，因此不需要標籤編碼。
- **Original Data**：顯示資料集矩陣的原始大小。在此實驗中（22800，24）表示22,800筆資料樣本和24個特徵（包括目標列）。
- **Missing Values**：當原始資料中缺少值時，它將顯示為True。對於此實驗，資料集中沒有缺少值。
- **Numeric Features**：推斷為數值的特徵數量。在此資料集中，將24個特徵中的14個推斷為數值。
- **Categorical Features**：推斷為分類的特徵數。在此資料集中，推斷出24個特徵中的9個是分類的。
- **Transformed Train Set**：顯示轉換後的訓練集矩陣的大小。訓練資料集中有15959筆樣本。此資料集分割使用預設值70/30(訓練/測試資料集)，可以使用“`train_size`”參數來更改。請注意，（22800，24）的原始形狀已針對轉換後的訓練集轉換為（15959，91），並且由於分類編碼，特徵數量已從24增加到91。
- **Transformed Test Set**：顯示轉換後的測試資料集的大小。測試資料集中有6841筆樣本。

## 7.0 比較所有模型(Comparing All Models)

建議在 `setup` 函式執行完成後，可以比較所有模型以評估性能，這是模型建立所建議起點（除非您確切地知道需要哪種模型，通常情況並非如此）。此功能訓練模型庫中的所有模型，並使用分層交叉驗證對度量進行評估，從而對它們進行評分。輸出結果會輸出一個分數網格，該網格顯示出平均準確度(average Accuracy)、接收者操作特徵曲線(receiver operating characteristic curve, ROC)下面積 (Area under the curve, AUC)、召回率(Recall)、精確度(Precision)、F1評分、Kappa係數和馬修斯相關係數 (Matthews correlation coefficient, MCC)（預設為10）以及訓練時間(training times)。

In [8]:

```
best_model = compare_models()
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
<b>ridge</b>	Ridge Classifier	0.8254	0.0000	0.3637	0.6913	0.4764	0.3836	0.4122	0.054
<b>lida</b>	Linear Discriminant Analysis	0.8247	0.7634	0.3755	0.6794	0.4835	0.3884	0.4132	0.309
<b>gbc</b>	Gradient Boosting Classifier	0.8226	0.7789	0.3551	0.6806	0.4664	0.3725	0.4010	5.650
<b>ada</b>	Ada Boost Classifier	0.8221	0.7697	0.3505	0.6811	0.4626	0.3690	0.3983	1.356
<b>catboost</b>	CatBoost Classifier	0.8218	0.7758	0.3669	0.6685	0.4736	0.3771	0.4018	13.718
<b>lightgbm</b>	Light Gradient Boosting Machine	0.8210	0.7750	0.3609	0.6679	0.4683	0.3721	0.3977	0.402
<b>rf</b>	Random Forest Classifier	0.8199	0.7598	0.3663	0.6601	0.4707	0.3727	0.3965	3.103
<b>xgboost</b>	Extreme Gradient Boosting	0.8160	0.7561	0.3629	0.6391	0.4626	0.3617	0.3829	5.677
<b>et</b>	Extra Trees Classifier	0.8092	0.7377	0.3677	0.6047	0.4571	0.3497	0.3657	2.452
<b>lr</b>	Logistic Regression	0.7814	0.6410	0.0003	0.1000	0.0006	0.0003	0.0034	1.271
<b>knn</b>	K Neighbors Classifier	0.7547	0.5939	0.1763	0.3719	0.2388	0.1145	0.1259	1.232
<b>dt</b>	Decision Tree Classifier	0.7293	0.6147	0.4104	0.3878	0.3986	0.2242	0.2245	0.395
<b>svm</b>	SVM - Linear Kernel	0.7277	0.0000	0.1017	0.1671	0.0984	0.0067	0.0075	0.522
<b>qda</b>	Quadratic Discriminant Analysis	0.4795	0.5267	0.6105	0.2493	0.3336	0.0435	0.0520	0.168
<b>nb</b>	Naive Bayes	0.3760	0.6442	0.8845	0.2441	0.3826	0.0608	0.1207	0.052

使用交叉驗證對15個模型進行了訓練和評估。上面顯示的分數網格指出了性能最高的指標，僅用於比較目的。預設情況下，網格使用“平均準確度(average Accuracy)”（從最高到最低）進行排序，可以通過傳遞排序參數來更改。例如 `compare_models ( sort = 'Recall' )` 將通過召回率(Recall)而不是平均準確度(average Accuracy)對網格進行排序。如果要將fold參數從預設值10更改為其他值，則可以使用fold參數。例如 `compare_models ( fold = 5 )` 將比較所有模型5次交叉驗證中的結果。而減少次數將縮短訓練時間。預設情況下，`compare_models` 函式根據預設的排序順序顯示性能最佳的模型，也可以使用n\_select參數來顯示前N個模型。

In [9]:

```
print(best_model)
```

```
RidgeClassifier(alpha=1.0, class_weight=None, copy_X=True, fit_intercept=True,
                 max_iter=None, normalize=False, random_state=123, solver='auto',
                 tol=0.001)
```

## 8.0 建立模型(Create a Model)

`create_model` 函式可透過“fold”參數設定的交叉驗證來訓練和評估模型。將輸出一個分數網格，以k-fold方式顯示平均準確度(average Accuracy) · ROC下面積 ( Area under the curve, AUC ) · 召回率(Recall) · 精確度(Precision) · F1評分 · Kappa係數和馬修斯相關係數(MCC)。

此次教學，將使用以下模型作為候選模型。這些選擇僅用於說明作法，並不意味著它們是此類資料的最佳選擇。

- 決策樹分類器(Decision Tree Classifier) ( 'dt' )
- K鄰近分類器(K Neighbors Classifier) ( 'knn' )
- 隨機森林分類器(Random Forest Classifier) ( 'rf' )

PyCaret的模型庫中有18個分類器。若要看所有分類器的列表，可使用 `models` 函式查看。

In [10]:

models()

Out[10]:

ID	Name	Reference	Turbo
lr	Logistic Regression	sklearn.linear_model._logistic.LogisticRegression	True
knn	K Neighbors Classifier	sklearn.neighbors._classification.KNeighborsCl...	True
nb	Naive Bayes	sklearn.naive_bayes.GaussianNB	True
dt	Decision Tree Classifier	sklearn.tree._classes.DecisionTreeClassifier	True
svm	SVM - Linear Kernel	sklearn.linear_model._stochastic_gradient.SGDC...	True
rbfsvm	SVM - Radial Kernel	sklearn.svm._classes.SVC	False
gpc	Gaussian Process Classifier	sklearn.gaussian_process._gpc.GaussianProcessC...	False
mlp	MLP Classifier	pycaret.internal.tunable.TunableMLPClassifier	False
ridge	Ridge Classifier	sklearn.linear_model._ridge.RidgeClassifier	True
rf	Random Forest Classifier	sklearn.ensemble._forest.RandomForestClassifier	True
qda	Quadratic Discriminant Analysis	sklearn.discriminant_analysis.QuadraticDiscrim...	True
ada	Ada Boost Classifier	sklearn.ensemble._weight_boosting.AdaBoostClas...	True
gbc	Gradient Boosting Classifier	sklearn.ensemble._gb.GradientBoostingClassifier	True
lda	Linear Discriminant Analysis	sklearn.discriminant_analysis.LinearDiscrimina...	True
et	Extra Trees Classifier	sklearn.ensemble._forest.ExtraTreesClassifier	True
xgboost	Extreme Gradient Boosting	xgboost.sklearn.XGBClassifier	True
lightgbm	Light Gradient Boosting Machine	lightgbm.sklearn.LGBMClassifier	True
catboost	CatBoost Classifier	catboost.core.CatBoostClassifier	True

## 8.1 決策樹分類器(Decision Tree Classifier)

In [11]:

```
dt = create_model('dt')
```

	<b>Accuracy</b>	<b>AUC</b>	<b>Recall</b>	<b>Prec.</b>	<b>F1</b>	<b>Kappa</b>	<b>MCC</b>
<b>0</b>	0.7343	0.6257	0.4327	0.4005	0.4160	0.2444	0.2447
<b>1</b>	0.7325	0.6277	0.4384	0.3984	0.4175	0.2443	0.2448
<b>2</b>	0.7431	0.6282	0.4241	0.4146	0.4193	0.2544	0.2544
<b>3</b>	0.7274	0.6151	0.4155	0.3856	0.4000	0.2240	0.2242
<b>4</b>	0.7187	0.6054	0.4040	0.3691	0.3858	0.2038	0.2042
<b>5</b>	0.7187	0.6014	0.3897	0.3656	0.3773	0.1958	0.1960
<b>6</b>	0.7206	0.6128	0.4212	0.3760	0.3973	0.2162	0.2168
<b>7</b>	0.7331	0.5986	0.3610	0.3830	0.3717	0.2024	0.2026
<b>8</b>	0.7206	0.6045	0.3983	0.3707	0.3840	0.2036	0.2038
<b>9</b>	0.7442	0.6272	0.4195	0.4148	0.4171	0.2533	0.2533
<b>Mean</b>	0.7293	0.6147	0.4104	0.3878	0.3986	0.2242	0.2245
<b>SD</b>	0.0092	0.0112	0.0218	0.0174	0.0173	0.0218	0.0218

In [12]:

```
#trained model object is stored in the variable 'dt'.
print(dt)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                      max_depth=None, max_features=None, max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, presort='deprecated',
                      random_state=123, splitter='best')
```

## 8.2 K鄰近分類器(K Neighbors Classifier)

In [13]:

```
knn = create_model('knn')
```

	<b>Accuracy</b>	<b>AUC</b>	<b>Recall</b>	<b>Prec.</b>	<b>F1</b>	<b>Kappa</b>	<b>MCC</b>
<b>0</b>	0.7469	0.6020	0.1920	0.3545	0.2491	0.1128	0.1204
<b>1</b>	0.7550	0.5894	0.2092	0.3883	0.2719	0.1402	0.1500
<b>2</b>	0.7506	0.5883	0.1576	0.3459	0.2165	0.0923	0.1024
<b>3</b>	0.7419	0.5818	0.1519	0.3136	0.2046	0.0723	0.0790
<b>4</b>	0.7563	0.5908	0.1490	0.3611	0.2110	0.0954	0.1085
<b>5</b>	0.7550	0.5997	0.1748	0.3720	0.2378	0.1139	0.1255
<b>6</b>	0.7638	0.5890	0.1891	0.4125	0.2593	0.1413	0.1565
<b>7</b>	0.7613	0.6240	0.1633	0.3904	0.2303	0.1163	0.1318
<b>8</b>	0.7619	0.5988	0.1862	0.4037	0.2549	0.1356	0.1500
<b>9</b>	0.7549	0.5756	0.1897	0.3771	0.2524	0.1246	0.1351
<b>Mean</b>	0.7547	0.5939	0.1763	0.3719	0.2388	0.1145	0.1259
<b>SD</b>	0.0065	0.0126	0.0191	0.0279	0.0214	0.0214	0.0230

## 8.3 隨機森林分類器(Random Forest Classifier)

In [14]:

```
rf = create_model('rf')
```

	<b>Accuracy</b>	<b>AUC</b>	<b>Recall</b>	<b>Prec.</b>	<b>F1</b>	<b>Kappa</b>	<b>MCC</b>
<b>0</b>	0.8133	0.7673	0.3610	0.6269	0.4582	0.3551	0.3749
<b>1</b>	0.8239	0.7615	0.3782	0.6735	0.4844	0.3882	0.4117
<b>2</b>	0.8258	0.7708	0.3467	0.7076	0.4654	0.3756	0.4098
<b>3</b>	0.8177	0.7605	0.3725	0.6436	0.4719	0.3710	0.3913
<b>4</b>	0.8208	0.7642	0.3725	0.6599	0.4762	0.3780	0.4006
<b>5</b>	0.8283	0.7638	0.3954	0.6866	0.5018	0.4070	0.4297
<b>6</b>	0.8127	0.7647	0.3582	0.6250	0.4554	0.3522	0.3721
<b>7</b>	0.8283	0.7390	0.3553	0.7168	0.4751	0.3861	0.4202
<b>8</b>	0.8108	0.7496	0.3610	0.6146	0.4549	0.3496	0.3678
<b>9</b>	0.8176	0.7565	0.3621	0.6462	0.4641	0.3645	0.3867
<b>Mean</b>	0.8199	0.7598	0.3663	0.6601	0.4707	0.3727	0.3965
<b>SD</b>	0.0062	0.0089	0.0131	0.0335	0.0139	0.0172	0.0202

請注意，所有模型的平均分數與在 `compare_models` 函式中顯示的分數匹配。這是因為在 `compare_models` 分數網格中顯示的指標是所有fold的平均分數。與 `compare_models` 函式類似，如果要將fold參數從預設值10更改為其他值，則可以使用fold參數。例如：`create_model('dt', fold = 5)` 將使用5個fold建立一個決策樹分類器。

## 9.0 調整模型(Tune a Model)

使用 `create_model` 函式建立模型時，使用預設的超參數來訓練模型。為了調整超參數，使用了 `tune_model` 函式。此功能使用“隨機網格搜索(Random Grid Search)”在預先定義的搜索空間上自動調整模型的超參數。所輸出的分數網格，以最佳形式顯示平均準確度(average Accuracy)、ROC下面積 ( Area under the curve, AUC )、召回率(Recall)、精確度(Precision)、F1評分、Kappa係數和馬修斯相關係數(MCC)。若要使用自行定義搜索網格，可以在 `tune_model` 函式中使用 `custom_grid` 參數（請參考下面的9.2 K鄰近分類器的調整）。

### 9.1 決策樹分類器(Decision Tree Classifier)

In [15]:

```
tuned_dt = tune_model(dt)
```

	<b>Accuracy</b>	<b>AUC</b>	<b>Recall</b>	<b>Prec.</b>	<b>F1</b>	<b>Kappa</b>	<b>MCC</b>
<b>0</b>	0.8177	0.7475	0.3095	0.6835	0.4260	0.3355	0.3728
<b>1</b>	0.8289	0.7711	0.3381	0.7375	0.4637	0.3782	0.4190
<b>2</b>	0.8208	0.7377	0.2894	0.7266	0.4139	0.3305	0.3796
<b>3</b>	0.8252	0.7580	0.3152	0.7333	0.4409	0.3563	0.4010
<b>4</b>	0.8195	0.7545	0.3095	0.6968	0.4286	0.3398	0.3794
<b>5</b>	0.8271	0.7509	0.3438	0.7186	0.4651	0.3769	0.4134
<b>6</b>	0.8195	0.7488	0.3123	0.6943	0.4308	0.3415	0.3801
<b>7</b>	0.8246	0.7529	0.2980	0.7482	0.4262	0.3446	0.3957
<b>8</b>	0.8195	0.7241	0.3123	0.6943	0.4308	0.3415	0.3801
<b>9</b>	0.8188	0.7378	0.3075	0.6903	0.4254	0.3362	0.3751
<b>Mean</b>	0.8222	0.7483	0.3136	0.7123	0.4352	0.3481	0.3896
<b>SD</b>	0.0037	0.0122	0.0156	0.0219	0.0159	0.0161	0.0158

In [16]:

```
#tuned model object is stored in the variable 'tuned_dt'.
print(tuned_dt)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='entropy',
                      max_depth=6, max_features=1.0, max_leaf_nodes=None,
                      min_impurity_decrease=0.002, min_impurity_split=None,
                      min_samples_leaf=5, min_samples_split=5,
                      min_weight_fraction_leaf=0.0, presort='deprecated',
                      random_state=123, splitter='best')
```

## 9.2 K鄰近分類器(K Neighbors Classifier)

In [17]:

```
import numpy as np
tuned_knn = tune_model(knn, custom_grid = {'n_neighbors' : np.arange(0,50,1)})
```

	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.7813	0.6482	0.0372	0.5000	0.0693	0.0402	0.0876
1	0.7807	0.6436	0.0315	0.4783	0.0591	0.0330	0.0759
2	0.7744	0.6563	0.0315	0.3333	0.0576	0.0206	0.0403
3	0.7845	0.6589	0.0659	0.5610	0.1179	0.0754	0.1345
4	0.7826	0.6645	0.0315	0.5500	0.0596	0.0368	0.0903
5	0.7794	0.6477	0.0544	0.4634	0.0974	0.0539	0.0961
6	0.7826	0.6278	0.0630	0.5238	0.1125	0.0688	0.1214
7	0.7751	0.6702	0.0372	0.3611	0.0675	0.0278	0.0523
8	0.7813	0.6409	0.0630	0.5000	0.1120	0.0662	0.1146
9	0.7881	0.6426	0.0661	0.6389	0.1198	0.0822	0.1548
Mean	0.7810	0.6501	0.0482	0.4910	0.0873	0.0505	0.0968
SD	0.0039	0.0119	0.0148	0.0861	0.0255	0.0206	0.0338

In [18]:

```
print(tuned_knn)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                      metric_params=None, n_jobs=-1, n_neighbors=42, p=2,
                      weights='uniform')
```

## 9.3 隨機森林分類器(Random Forest Classifier)

In [19]:

```
tuned_rf = tune_model(rf)
```

	<b>Accuracy</b>	<b>AUC</b>	<b>Recall</b>	<b>Prec.</b>	<b>F1</b>	<b>Kappa</b>	<b>MCC</b>
<b>0</b>	0.8158	0.7508	0.3181	0.6647	0.4302	0.3363	0.3689
<b>1</b>	0.8283	0.7675	0.3295	0.7419	0.4563	0.3719	0.4152
<b>2</b>	0.8139	0.7337	0.3181	0.6529	0.4277	0.3321	0.3628
<b>3</b>	0.8246	0.7588	0.3095	0.7347	0.4355	0.3514	0.3976
<b>4</b>	0.8170	0.7567	0.3438	0.6557	0.4511	0.3539	0.3805
<b>5</b>	0.8258	0.7506	0.3324	0.7205	0.4549	0.3676	0.4067
<b>6</b>	0.8170	0.7530	0.3324	0.6629	0.4427	0.3474	0.3771
<b>7</b>	0.8221	0.7507	0.3381	0.6901	0.4538	0.3621	0.3951
<b>8</b>	0.8177	0.7201	0.2980	0.6933	0.4168	0.3286	0.3699
<b>9</b>	0.8207	0.7484	0.3132	0.6987	0.4325	0.3439	0.3831
<b>Mean</b>	0.8203	0.7490	0.3233	0.6915	0.4402	0.3495	0.3857
<b>SD</b>	0.0045	0.0126	0.0135	0.0310	0.0129	0.0140	0.0165

在預設情況下，使用 `tune_model` 函式將根據平均準確率來最佳化模型超參數，但是可以使用 `optimize` 參數對其進行更改。例如：`tune_model(dt, optimize='AUC')` 將搜索決策樹分類器的超參數，該超參數將以 AUC 來最佳化模型的超參數，而不是平均準確性。在最終確定最佳生產模型時，度量標準不一定是唯一標準。還可以考慮的因素包括訓練時間， $k$ -fold 的標準差等。在本次教學中，將調整隨機森林分類器的 `tuned_rf` 模型變數作為剩餘部分的最佳模型。

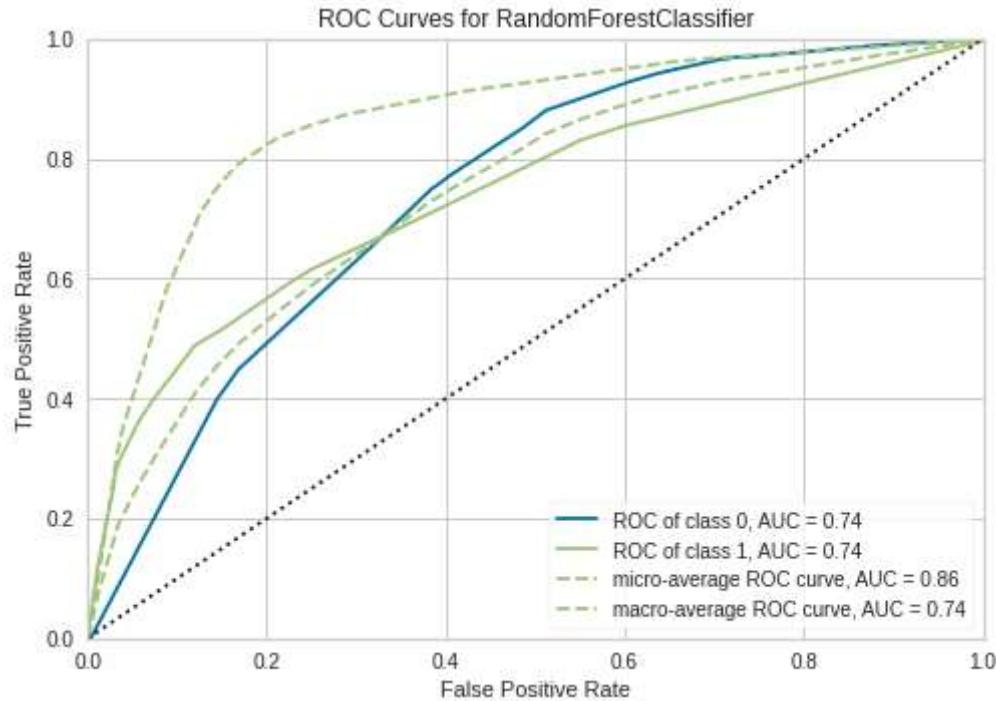
## 10.0 各種圖形分析模型性能(Plot a Model)

在模型完成之前，`plot_model` 函式可用於分析不同方面的性能，例如 AUC、混淆矩陣、決策邊界等。共有 [15種不同的繪圖](https://pycaret.org/plot-model/) (<https://pycaret.org/plot-model/>) 可用。

### 10.1 AUC Plot

In [20]:

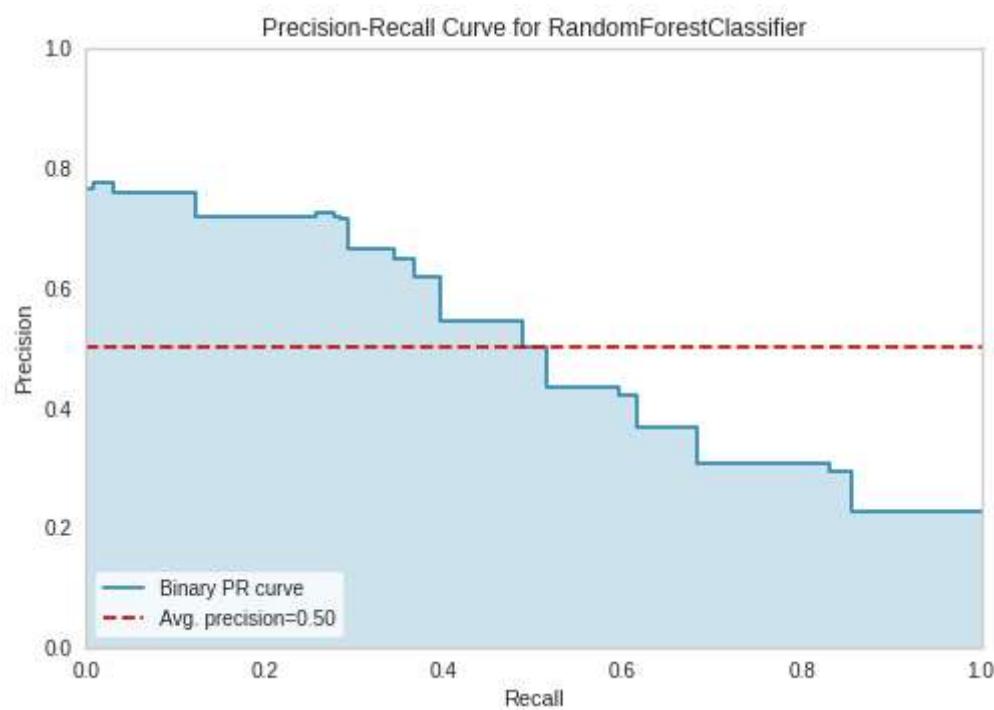
```
plot_model(tuned_rf, plot = 'auc')
```



## 10.2 Precision-Recall Curve

In [21]:

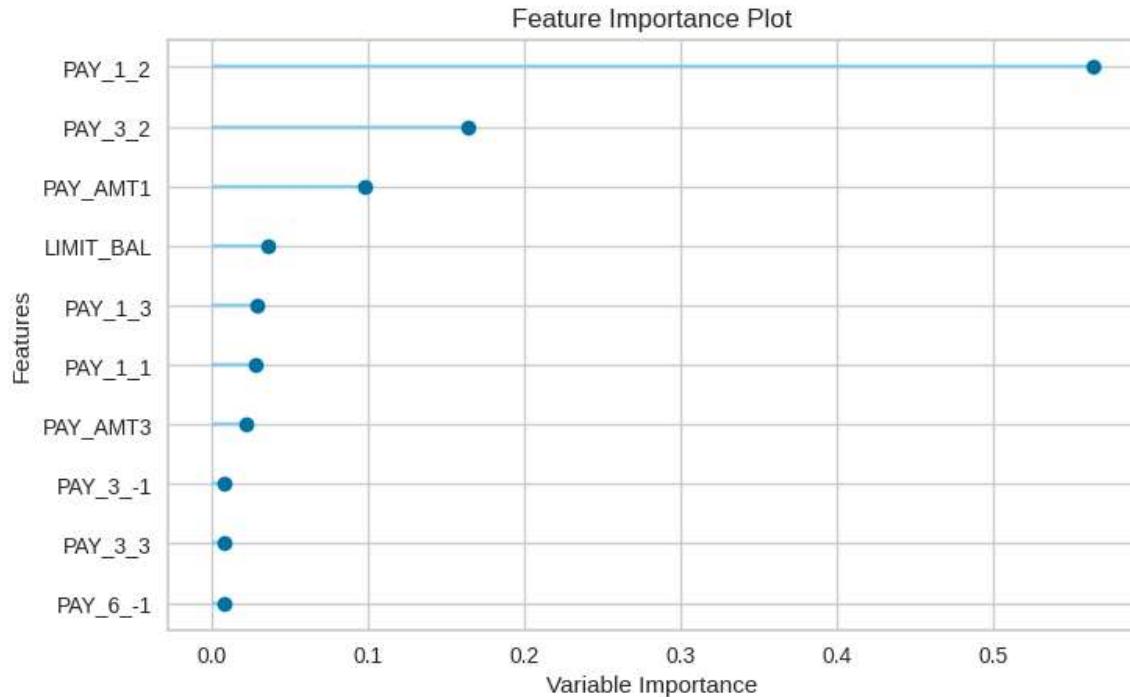
```
plot_model(tuned_rf, plot = 'pr')
```



## 10.3 Feature Importance Plot

In [22]:

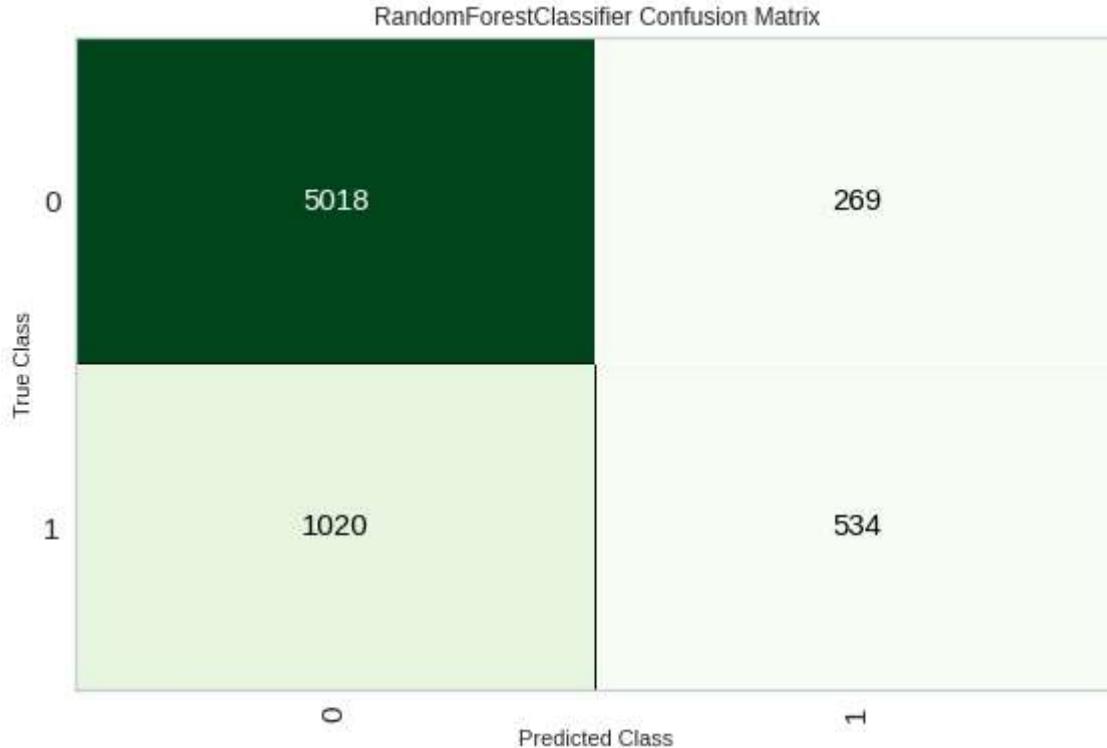
```
plot_model(tuned_rf, plot='feature')
```



## 10.4 Confusion Matrix

In [23]:

```
plot_model(tuned_rf, plot = 'confusion_matrix')
```



分析模型性能的另一種方法是使用 `evaluate_model` 函式，該函式顯示給定模型的所有可用圖形的使用者界面。它在內部是使用 `plot_model` 函式。

In [24]:

```
evaluate_model(tuned_rf)
```

## 11.0 預測測試資料樣本(Predict on test Sample)

在最終確定模型之前，建議透過預測測試資料集並查看評估指標來執行最終檢查。如果看一下上面第6節中的資訊網格，將看到30% ( 6,841筆資料樣本 ) 的資料已被分離為測試資料樣本。上面看到的所有評估指標都是使用訓練集70%(15,959筆資料樣本)的交叉驗證結果。現在，使用儲存在tuned\_rf變數中的最終訓練模型，將針對測試樣本進行預測並評估指標。

In [25]:

```
predict_model(tuned_rf);
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	Random Forest Classifier	0.8116	0.7407	0.3436	0.665	0.4531	0.353	0.3811

測試資料集的準確率為 **0.8135**，而 tuned\_rf 變數的結果準確率為 **0.8225**（在上面的9.3節中）。這不是顯著差異。如果測試和訓練資料集預測結果之間存在較大差異，則通常表示過度擬合(over-fitting)，但也可能是由於其他因素所致，需要進一步研究。在這種情況下，將繼續進行模型的最終確定和對看不見的數據進行預測（一開始就分離出的5%，從未暴露給PyCaret的實驗過程）。

## 12.0 部署模型確定(Finalize Model for Deployment)

模型確定是模型訓練實驗的最後一步。在PyCaret中，正常的機器學習工作流程從 setup 函式開始，然後使用 compare\_models 函式比較所有模型，並從一些候選模型中篩選出待調整模型，以執行建立模型的方法，例如超參數調整，集合，堆疊等。該工作流程最終將引導您找到最佳模型，用於對新數據和看不見的數據進行預測。 finalize\_model 函式將模型適應到完整的資料集上，包括測試資料樣本（在這種情況下為30%）。該功能的目的是在將模型部署到生產中之前，在完整的資料集中對其進行訓練。

In [26]:

```
final_rf = finalize_model(tuned_rf)
```

In [27]:

```
#Final Random Forest model parameters for deployment
print(final_rf)
```

```
RandomForestClassifier(bootstrap=False, ccp_alpha=0.0, class_weight={},
                      criterion='entropy', max_depth=5, max_features=1.0,
                      max_leaf_nodes=None, max_samples=None,
                      min_impurity_decrease=0.0002, min_impurity_split=None,
                      min_samples_leaf=5, min_samples_split=10,
                      min_weight_fraction_leaf=0.0, n_estimators=150,
                      n_jobs=-1, oob_score=False, random_state=123, verbose=0,
                      warm_start=False)
```

In [28]:

```
predict_model(final_rf);
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	Random Forest Classifier	0.8184	0.7526	0.3533	0.6985	0.4692	0.3736	0.4053

請注意，為何，AUC如何從 **0.7563** 增加到 **0.8222**。這是因為 `finalize_model` 函式完成模型確定後，將使用包括測試資料集的整個資料集進行訓練。而 `finalize_model` 函式之後將模型用於對保留集的預測。

## 13.0 預測未見過的資料集(Predict on unseen data)

`predict_model` 函式還用於在未見過的資料集上進行預測。與上面第11節的唯一區別，是這次我們將傳遞 `data_unseen`參數。`data_unseen`是在教學開始時建立的變數，包含從未公開給PyCaret的原始數據集的5% ( 1200個樣本 )。

In [29]:

```
unseen_predictions = predict_model(final_rf, data=data_unseen)
unseen_predictions.head()
```

Out[29]:

	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_1	PAY_2	PAY_3	PAY_4	PAY_5
0	100000	2	2	2	23	0	-1	-1	0	0
1	380000	1	2	2	32	-1	-1	-1	-1	-1
2	200000	2	2	1	32	-1	-1	-1	-1	2
3	200000	1	1	1	53	2	2	2	2	2
4	240000	1	1	2	41	1	-1	-1	0	0

Label和Score變數將自動增加到`data_unseen`資料集中。`label`是預測結果，分數是預測的機率。注意，將預測結果連接到原始資料集，而所有轉換都在後台自動執行。此外，還可以檢查指標，因為有實際的目標預設值。為此，將使用`pycaret.utils`模組來計算平均準確率。可參考下列範例：

In [30]:

```
from pycaret.utils import check_metric
check_metric(unseen_predictions.default, unseen_predictions.Label, 'Accuracy')
```

Out[30]:

0.8167

## 14.0 模型儲存(Saving the model)

現在，透過完成tuned\_rf變數模型的儲存來完成實驗。該模型現在儲存在final\_rf變數中。還使用了儲存在final\_rf中的模型來預測data\_unseen。這將帶到實驗的終點，但是仍然有一個問題要問：當有更多新資料可以預測時會發生什麼？是否需要重新進行整個實驗？答案是不需要，PyCaret內建的函式save\_model允許將模型與整個轉換管道一起保存以備後用。

In [31]:

```
save_model(final_rf, 'Final RF Model 08Feb2020')
```

Transformation Pipeline and Model Successfully Saved

Out[31]:

```
(Pipeline(memory=None,
      steps=[('dtyp...',  

              DataTypes_Auto_infer(categorical_features=[],  

                                    display_types=True, features_todrop  

= [],  

                                    id_columns=[],  

                                    ml_usecase='classification',  

                                    numerical_features=[], target='defa  

ult',  

                                    time_features=[])),  

      ('imputer',  

       Simple_Imputer(categorical_strategy='not_available',  

                      fill_value_categorical=None,  

                      fill_value_numerical=None,  

                      numeric_st...  

RandomForestClassifier(bootstrap=False, ccp_alpha=0.0,  

                      class_weight={}, criterion='entro  

py',  

                      max_depth=5, max_features=1.0,  

                      max_leaf_nodes=None, max_samples=  

None,  

                      min_impurity_decrease=0.0002,  

                      min_impurity_split=None,  

                      min_samples_leaf=5,  

                      min_samples_split=10,  

                      min_weight_fraction_leaf=0.0,  

                      n_estimators=150, n_jobs=-1,  

                      oob_score=False, random_state=12  

3,  

                      verbose=0, warm_start=False)]],  

      verbose=False), 'Final RF Model 08Feb2020.pkl')
```

(提示：保存模型時，最好在文件名中使用日期，這對於版本控制很有用。)

## 15.0 載入和保存模型>Loading the saved model)

為了將來在相同或替代環境中載入保存的模型，將使用PyCaret的load\_model函式，輕鬆地將保存的模型應用於之前保留的新資料以進行預測。

In [32]:

```
saved_final_rf = load_model('Final RF Model 08Feb2020')
```

Transformation Pipeline and Model Successfully Loaded

將模型載入到環境中後，可以使用相同的 `predict_model` 函式簡單地使用它來預測任何新 資料。之後，應用載入的模型來預測與上文第13節中使用的相同的`data_unseen`。

In [33]:

```
new_prediction = predict_model(saved_final_rf, data=data_unseen)
```

In [34]:

```
new_prediction.head()
```

Out[34]:

	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_1	PAY_2	PAY_3	PAY_4	PAY_5
0	100000	2	2	2	23	0	-1	-1	0	0
1	380000	1	2	2	32	-1	-1	-1	-1	-1
2	200000	2	2	1	32	-1	-1	-1	-1	2
3	200000	1	1	1	53	2	2	2	2	2
4	240000	1	1	2	41	1	-1	-1	0	0

注意，`unseen_predictions`和`new_prediction`帶有預測結果的資料集變數的結果相同。

In [35]:

```
from pycaret.utils import check_metric
check_metric(new_prediction.default, new_prediction.Label, 'Accuracy')
```

Out[35]:

0.8167

## 16.0 總結

本次教學涵蓋了從獲取資料，前處理，模型訓練，超參數調整，預測和保存模型以供以後使用的整個機器學習管道。且用不到10條程式碼就可以完成所有這些步驟，而這些命令還很容易記住，例如 `create_model`，`tune_model` 和 `compare_models` 函式。在沒有PyCaret的情況下重新實作整個實驗在大多數的函示庫需花費超過100行程式碼。

此教學僅介紹了 `pycaret.classification` 模組使用的基礎知識。在接下來的教學中，我們將更深入地進行高級前處理，組合，通用堆疊和其他技術，這些技術可讓您完全自行定義機器學習管道，並且是任何資料科學家都必須知道的。

接下來的教學文件請參考 [二元分類教學 \( CLF102 \) - 中級](#)

(<https://github.com/pycaret/pycaret/blob/master/tutorials/Binary%20Classification%20Tutorial%20Level%20Intermediate%20CLF102.ipynb>)

