第13章　員工流失率預測

第13章　員工流失率預測

- 員工流失率的案例示範
- 資料載入與檢查
- 資料探索
- 資料切割與資料預處理
- 用網格搜尋探索各種模型的最佳結果
- PRC和 ROC圖

- 員工是公司最重要的資產，好的員工能為公司帶來獲利和成長。我們能不能用機器學習的方法，來預測哪個員工即將要離職？本章將透過機器學習方式來進行員工流失的預測。數據來源為Kaggle 的Human Resources Analytics。

本章需先載入的資料和欄位說明：

- satisfaction_level：對公司的滿意程度
- last_evaluation：上一次的公司考評
- number_projects：負責專案的數量
- average_monthly_hours：平均每月工時
- time_spend_company：在公司待了幾年
- Work_accident：是否曾有工作事故
- left：是否離職（目標值）
- promotion_last_5years：最近5 年是否有晉升
- sales：在哪個部門
- salary：薪資水平

範例13-1 載入資料和欄位

程式碼

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
%config InlineBackend.figure_format = 'retina'
import warnings
warnings.filterwarnings('ignore')
df = pd.read_csv('HR_comma_sep.csv')
df.head()
```

執行結果

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion_last_5years	sales	salary
0	0.38	0.53	2	157	3	0	1	0	sales	low
1	0.80	0.86	5	262	6	0	1	0	sales	medium
2	0.11	0.88	7	272	4	0	1	0	sales	medium
3	0.72	0.87	5	223	5	0	1	0	sales	low
4	0.37	0.52	2	159	3	0	1	0	sales	low

範例13-2 檢視資料的個數和基本形態

程式碼

```
df.info()
```

執行結果

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14999 entries, 0 to 14998
Data columns (total 10 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   satisfaction_level     14999 non-null  float64
 1   last_evaluation        14999 non-null  float64
 2   number_project         14999 non-null  int64
 3   average_montly_hours   14999 non-null  int64
 4   time_spend_company     14999 non-null  int64
 5   Work_accident          14999 non-null  int64
 6   left                   14999 non-null  int64
 7   promotion_last_5years  14999 non-null  int64
 8   sales                  14999 non-null  object
 9   salary                 14999 non-null  object
dtypes: float64(2), int64(6), object(2)
memory usage: 1.1+ MB
```

範例13-3 進一步檢視sales 欄位

程式碼

```
df['sales'].value_counts()
```

執行結果

```
sales          4140
technical      2720
support        2229
IT             1227
product_mng     902
marketing       858
RandD           787
accounting      767
hr              739
management      630
Name: sales, dtype: int64
```

範例13-4 檢視salary 欄位

程式碼

```
df['salary'].value_counts()
```

執行結果

```
low        7316
medium     6446
high       1237
Name: salary, dtype: int64
```

範例13-5 檢視Work_accident 欄位

程式碼

```
df['Work_accident'].value_counts()
```

執行結果

```
0    12830
1     2169
Name: Work_accident, dtype: int64
```

範例13-6 檢視目標欄位left

程式碼

```
size = df['left'].value_counts()
pct = df['left'].value_counts(normalize=True).round(2)
pd.DataFrame(zip(size, pct), columns=['次數', '百分比'])
```

執行結果

	次數	百分比
0	11428	0.76
1	3571	0.24

範例13-16 資料切割

程式碼

```
X = df.drop('left', axis=1)
y = df['left']
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.3, random_state=42)
```

範例13-17 將X 欄位再細分成數值和類別

程式碼

```
X_col_cat = X.select_dtypes(include = 'object').columns
X_col_num = X.select_dtypes(exclude = 'object').columns
print(f'類別型資料欄位：{X_col_cat}')
print(f'數值型資料欄位：{X_col_num}')
```

執行結果

```
類別型資料欄位：Index(['sales', 'salary'], dtype='object')
數值型資料欄位：Index(['satisfaction_level', 'last_evaluation',
'number_project',
       'average_montly_hours', 'time_spend_company', 'Work_accident',
       'promotion_last_5years'],
      dtype='object')
```

範例13-18　建構數值和類別兩個管道器，並整合到合併器

程式碼

```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
data_pl = ColumnTransformer([
    ('num', StandardScaler(), X_col_num),
    ('cat', OneHotEncoder(), X_col_cat)
])
```

範例13-19 求出「基礎」的預測結果

程式碼

```
from sklearn.dummy import DummyClassifier
from sklearn.metrics import confusion_matrix,
classification_report,accuracy_score
dmy = DummyClassifier(strategy='most_frequent')
dmy.fit(X_train, y_train)
dmy.score(X_train, y_train)
y_pred = dmy.predict(X_test)
print('正確率：', accuracy_score(y_test, y_pred).round(2))
print('混亂矩陣')
print(confusion_matrix(y_test, y_pred))
print('綜合報告')
print(classification_report(y_test, y_pred))
```

執行結果

正確率： 0.76

混亂矩陣

```
[[3428    0]
 [1072    0]]
```

綜合報告

```
              precision    recall  f1-score   support

           0       0.76      1.00      0.86      3428
           1       0.00      0.00      0.00      1072

   micro avg       0.76      0.76      0.76      4500
   macro avg       0.38      0.50      0.43      4500
weighted avg       0.58      0.76      0.66      4500
```

範例13-20　用GridSearchCV來挑選最佳結果 (I)

程式碼

```
# 載入所有模型
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier,
                              BaggingClassifier, AdaBoostClassifier
from xgboost import XGBClassifier
# 載入 Pipeline ，PCA 和 GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
```

範例13-20 用GridSearchCV來挑選最佳結果 (I)

承接上一頁

```
model_pl = Pipeline([
    ('preprocess', data_pl),
    ('model', LogisticRegression())
])
param_grid = {'model':[LogisticRegression(), SVC(),
              KNeighborsClassifier(),
              DecisionTreeClassifier(max_depth=10)]}
gs = GridSearchCV(model_pl, param_grid=param_grid,
                  cv=5, return_train_score=True)
gs.fit(X_train, y_train)
score = gs.best_estimator_.score(X_test, y_test)
```

範例13-20 用GridSearchCV來挑選最佳結果(I)

承接上一頁

```
print('最佳預測參數', gs.best_params_)
print('訓練集交叉驗證的最佳結果', gs.best_score_.round(3))
print('測試集的結果', score.round(3))
y_pred = gs.best_estimator_.predict(X_test)
print('混亂矩陣\n',confusion_matrix(y_test, y_pred))
```

執行結果

```
最佳預測參數 {'model': DecisionTreeClassifier(class_weight=None,
criterion='gini', max_depth=10,
            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=False, random_
state=None,
            splitter='best')}
訓練集交叉驗證的最佳結果 0.978
測試集的結果 0.977
混亂矩陣
 [[3394   34]
 [  69 1003]]
```

範例13-21 用GridSearchCV來挑選最佳結果(II)

程式碼

```
model_pl = Pipeline([('preprocess', data_pl), ('model',
LogisticRegression())])
np.random.seed(42)
param_grid = {'model':[RandomForestClassifier(),
AdaBoostClassifier(),
                    BaggingClassifier(), XGBClassifier()]}
gs = GridSearchCV(model_pl, param_grid=param_grid,
                cv=5, return_train_score=True)
gs.fit(X_train, y_train)
score = gs.best_estimator_.score(X_test, y_test)
```

範例13-21 用GridSearchCV來挑選最佳結果 (II)

承接上一頁

```
print('最佳預測參數', gs.best_params_)
print('訓練集交叉驗證的最佳結果', gs.best_score_.round(3))
print('測試集的結果', score.round(3))
y_pred = gs.best_estimator_.predict(X_test)
print('混亂矩陣\n',confusion_matrix(y_test, y_pred))
```

執行結果

最佳預測參數 {'model': BaggingClassifier(base_estimator=None,
bootstrap=True,
          bootstrap_features=False, max_features=1.0, max_
samples=1.0,
          n_estimators=10, n_jobs=None, oob_score=False, random_
state=None,
          verbose=0, warm_start=False)}
訓練集交叉驗證的最佳結果 0.986
測試集的結果 0.984
混亂矩陣

```
[[3407   21]
 [  50 1022]]
```

範例13-23 繪製ROC圖

程式碼

```
from sklearn.metrics import roc_curve,
roc_auc_score
y_pred_proba =
model_pl_rf.predict_proba(X_test)[:,1]
fpr, tpr, thresholds = roc_curve(y_test,
                                 y_pred_proba)
plt.plot(fpr, tpr)
plt.xlim(-0.01,1)
plt.ylim(0,1.01)
plt.plot([0,1],[0,1], ls='--')
roc_auc_score(y_test,
model_pl_rf.predict_proba(X_test)[:,1])
```

執行結果

0.986211826601647