員工流失率預測

```
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()
```

```
df.info()
```

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

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

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

```
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'])
```

```
sns.pairplot(df, hue='left', diag_kws={'bw':0.1});
sns.heatmap(df.corr().round(2), annot=True, cmap='coolwarm');
```

```
df.drop('left', axis=1).corrwith(df['left']).round(2)
```

```
df_left_salary = df.groupby(['left','salary']).size().unstack(1)
df_left_salary = df_left_salary[['low', 'medium', 'high']]
df_left_salary
```

```
df_left_salary/df_left_salary.sum()
```

```
df['time_spend_company'].value_counts().sort_index().\
plot(kind='bar', grid=True);
```

```
df_left_time = df.groupby(['left','time_spend_company']).size().unstack(0)
df_left_time.plot(kind='bar');
```

```
df.groupby(['left','sales']).size().unstack(0).plot(kind='bar');
```

```
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)
```

```
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}')
```

```
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)
])
```

```
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))
```

```
# 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
# 載作 Pipeline，PCA 和 GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV

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)
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))



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)
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))
```

```
model_pl_rf = Pipeline([
    ('preprocess', data_pl),
    ('model', RandomForestClassifier(random_state=42))
])
model_pl_rf.fit(X_train, y_train)
imp = model_pl_rf.named_steps['model'].feature_importances_
feature_names = model_pl_rf.named_steps['preprocess'].\
named_transformers_['cat'].get_feature_names(['sales','salary'])
cols = X_col_num.tolist() + feature_names.tolist()
pd.DataFrame(zip(cols, imp), columns=['欄位', '係數']).\
sort_values(by='係數', ascending=False).head()
```

```
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])
```

```python
from sklearn.metrics import precision_recall_curve
fpr, tpr, thresholds = precision_recall_curve(y_test,
                                              y_pred_proba)
plt.plot(fpr, tpr);
```