ADP_Red\ml\2.classification.py

```python
# %% 5. Machine Learning - Classification
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

import scipy.stats as stats

# %% 1. 데이터 수집
df = pd.read_csv('../../ADP_Python/data/bodyPerformance.csv')

print(df.shape)
print(df.info())

# %% Check binary variable
for i, var in enumerate(df.columns):
    print(i, var, len(df[var].unique()))

# %% 2. 데이터 결측치 보정
print(df.isna().sum())

# # 결측치 제거
# missing = ['bill_length_mm', 'bill_depth_mm', 'flipper_length_mm', 'body_mass_g']

# for i in missing:
#     df[i] = df[i].fillna(df[i].median())
# df['sex'] = df['sex'].fillna('Male')


# %% 3. 라벨 인코딩
from sklearn.preprocessing import LabelEncoder

label = ['gender', 'class']

df[label] = df[label].apply(LabelEncoder().fit_transform)
# df['gender'] = np.where(df['class']=='M', 0, 1)
# df['class'] = np.where(df['class']=='A', 1, 0)

print(df.info())
print(df.head())

# # %% data visualization
# fig, axes = plt.subplots(nrows=3, ncols=4, constrained_layout=True)

# for i, var in enumerate(df.columns):
#     row, col = i//4, i%4
#     sns.regplot(df, x=var, y='class',
#                 marker='o', ax=axes[row][col])

# plt.show()

# #
# from pandas.plotting import scatter_matrix

# scatter_matrix(df)
# plt.show

# %% 4. 데이터타입, 더미변환 (One-Hot Encoding)
# import pandas as pd

# category  = ['gender', 'class']
# for i in category:
#     df[i] = df[i].astype('category')
# df = pd.get_dummies(df)
# df.head()

# %% 5. 파생변수 생성
# df['body_mass_g_qcut'] = pd.qcut(df['body_mass_g'], 5, labels=False)


# %% 6. 정규화 또는 스케일 작업
# from sklearn.preprocessing import StandardScaler, MinMaxScaler

# scaling_vars = ['age', 'height_cm', 'weight_kg', 'body fat_%', 'diastolic', 'systolic', 'gripForce', 'sit and bend forward_cm', 'sit-ups counts', 'broad jump_cm']
# scaler = StandardScaler()
# # scaler = MinMaxScaler()
# scaler.fit(df[scaling_vars])

# df[scaling_vars] = scaler.transform(df[scaling_vars])


# %% 7. 데이터 분리
from sklearn.model_selection import train_test_split

X = df.iloc[:, :-1]
y = df.iloc[:,-1]

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, stratify=y, random_state=1)

X_train = np.array(X_train)
X_test = np.array(X_test)
y_train = np.array(y_train)
y_test = np.array(y_test)


print('X_train: ', X_train.shape)
print('X_test: ', X_test.shape)
print('y_train: ', y_train.shape)
print('y_test: ', y_test.shape)


# %% 8. 모델 학습
from sklearn.linear_model import LogisticRegression

# model = LogisticRegression()                                        # Logistic Regression
model = LogisticRegression(multi_class='multinomial', solver='lbfgs')   # Softmax Regression
model.fit(X_train, y_train)


# %% 9. 모델 학습 (2)
# import statsmodels.api as sm
# import statsmodels.formula.api as smf
# dv = 'class'

# model = smf.glm(
#     data=df, formula="class~age+height_cm+weight_kg",
#     family=sm.families.Binomial()
# ).fit()

# model.summary()

# %% 10. 앙상블

# %% 11. 모델 평가
from sklearn.metrics import mean_absolute_error, mean_squared_error
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.metrics import confusion_matrix, classification_report

pred = model.predict(X_test)
pred_proba = model.predict_proba(X_test)

print(f'MAE {mean_absolute_error(pred, y_test)}')
print(f'MSE {mean_squared_error(pred, y_test):.2f}')
print(f'RMSE {np.sqrt(mean_squared_error(pred, y_test)):.2f}')

print(f'혼동행렬: {confusion_matrix(pred, y_test)}')

print(f'정확도: {accuracy_score(pred, y_test) * 100 :.2f} % ')
# print(f'정밀도: {precision_score(pred, y_test) * 100 :.2f} % ')
# print(f'재현율: {recall_score(pred, y_test) * 100 :.2f} % ')
# print(f'F1   : {f1_score(pred, y_test) * 100 :.2f} % ')

# ROC Curve
# from sklearn.metrics import RocCurveDisplay

# RocCurveDisplay.from_estimator(model, X_test, y_test)
# plt.show()


# %% 12. 하이퍼파라미터 튜닝
# from sklearn.model_selection import GridSearchCV

# parameters = {'n_estimators':[50,100], 'max_depth':[4,6]}
# model4 = RandomForestClassifier()
# clf = GridSearchCV(estimator=model4, param_grid=parameters, cv=3)
# clf.fit(X_train, y_train)

# print(f'Best Parameter: {clf.best_params_}')


# %% 13. 예측값 저장
# Save Output
output = pd.DataFrame({'id': y_test.index, 'pred': pred})
output.to_csv('output.csv', index=False)

# Check Output
check = pd.read_csv('output.csv')
check.head()


# %% References
# - [[딥러닝] 로지스틱 회귀](https://circle-square.tistory.com/94)
# - [Logistic Regression in Python with statsmodels](https://www.andrewvillazon.com/logistic-regression-python-statsmodels/)
```