ADP_Red\stats\5.classification.py

```python
# 5. Classification
# - Logistic/Softmax Regression
# - Sigmoid Function (Logistic Function)
# - Odds

# %% 5-1. Logistic/Softmax Regression (Scikit-Learn)
import numpy as np
from sklearn import datasets

# Load Dataset
iris = datasets.load_iris()
X = iris.data[:, [2,3]]
y = iris.target

# Train-Test Split
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=1, stratify=y     # stratification: 훈련데이터와 테스트데이터의 클래스 레이블 비율 동일하게
)

# Scaling
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
sc.fit(X_train)

X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)   # 훈련데이터의 mu와 sigma로 scaling

# Model Fitting
from sklearn.linear_model import LogisticRegression

lr = LogisticRegression(C=100.0, random_state=1)
lr.fit(X_train_std, y_train)

lr.predict_proba(X_test_std[:3, :])
lr.predict(X_test_std[:3, :])









# %%
# %% 1. 데이터 수집
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

import scipy.stats as stats

# Load Dataset
```

```python
df = pd.read_csv('../ADP_Python/data/bodyPerformance.csv')

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

# %% 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)


# %% 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)


# %% 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 = ['bill_length_mm', 'bill_depth_mm', 'flipper_length_mm', 'body_mass_g']
# 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_train, X_test, y_train, y_test = train_test_split(
    df.iloc[:, :-1], df.iloc[:,-1], test_size=0.3, stratify=df.iloc[:,-1], random_state=1)

X_train = np.array(X_train)
X_test = np.array(X_test)
y_train = np.array(y_train).reshape(-1,1)
y_test = np.array(y_test).reshape(-1,1)


```

```
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

lr = LogisticRegression(C=100.0, random_state=1)
lr.fit(X_train, y_train)

lr.predict_proba(X_test[:3, :])
lr.predict(X_test[:3, :])


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

pred = lr.predict(X_test)

print(f'Model Accurary {accuracy_score(y_test, pred)}')
print()

# %% 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': pred3})
output.to_csv('00300.csv', index=False)

# Check Output
check = pd.read_csv('00300.csv')
check.head()

```