ADP_Red\ml\4.dt_ensemble.py

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

import scipy.stats as stats

# %% 0. 합성 데이터
X = np.sort(5 * np.random.rand(400,1), axis=0)
T = np.linspace(0, 5, 500)[:, np.newaxis]
y = np.sin(X).ravel()

y[::1] += 1 * (0.5 - np.random.rand(400))
plt.scatter(X, y, s=20, label='data')
plt.show()

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

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

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

# Check data summary
print(df.describe())


# %% 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

continuous_vars = ['credit.duration.months', 'credit.amount', 'age']
discrete_vars = ['account.balance',
       'previous.credit.payment.status', 'credit.purpose',
       'savings', 'employment.duration', 'installment.rate', 'marital.status',
       'guarantor', 'residence.duration', 'current.assets',
       'other.credits', 'apartment.type', 'bank.credits', 'occupation',
       'dependents', 'telephone', 'foreign.worker']

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

```

```
58  print(df.info())
59  print(df.head())
60
61  # #
62  # from pandas.plotting import scatter_matrix
63
64  # scatter_matrix(df)
65  # plt.show
66
67  # %% 4. 데이터타입, 더미변환 (One-Hot Encoding)
68  # import pandas as pd
69
70  # category  = ['gender', 'class']
71  # for i in category:
72  #     df[i] = df[i].astype('category')
73  # df = pd.get_dummies(df)
74  # df.head()
75
76  # %% 5. 파생변수 생성
77  # df['body_mass_g_qcut'] = pd.qcut(df['body_mass_g'], 5, labels=False)
78
79
80  # %% 6. 정규화 또는 스케일 작업
81  from sklearn.preprocessing import StandardScaler, MinMaxScaler
82
83  scaling_vars = continuous_vars
84  # scaler = StandardScaler()
85  scaler = MinMaxScaler()
86  scaler.fit(df[scaling_vars])
87
88  df[scaling_vars] = scaler.transform(df[scaling_vars])
89
90  # Boxplot for scaling check
91  sns.boxplot(df)
92  plt.tight_layout()
93  plt.show()
94
95
96  # %% 7. 데이터 분리
97  from sklearn.model_selection import train_test_split
98
99  # X = df.iloc[:, 1:]
100 # y = df.iloc[:,0]
101
102 X_train, X_test, y_train, y_test = train_test_split(
103     X, y, test_size=0.3, random_state=1,
104     # stratify=y
105     )
106
107 X_train = np.array(X_train)
108 X_test = np.array(X_test)
109 y_train = np.array(y_train)
110 y_test = np.array(y_test)
111
112
113 print('X_train: ', X_train.shape)
114 print('X_test: ', X_test.shape)
115 print('y_train: ', y_train.shape)
116 print('y_test: ', y_test.shape)
117
```

```python

# %% 8. 모델 학습
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor
from sklearn.linear_model import LogisticRegression

# model = DecisionTreeClassifier(max_depth=5)                                    # Decision Tree Classification
model = DecisionTreeRegressor(max_depth=5)                                    # Decision Tree Regression

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


# %% 9. 모델 학습 (2)
# (Decision Tree) Feature Importances
df_feature = pd.DataFrame([X.columns, model.feature_importances_]).T
df_feature.columns = ['feature_nm', 'importances']
print(df_feature)


# %% 10. 앙상블

# %% 11. 모델 평가
from sklearn.metrics import mean_absolute_error, mean_squared_error
from sklearn.metrics import r2_score
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, classification_report

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

# X_test_poly = poly_reg.fit_transform(X_test)  # Multinomial Regression
# pred = model.predict(X_test_poly)

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}')

# Metrics For Regression
print(f'R2 Score: {r2_score(pred, y_test):.2f}')

# # Metrics For Classification
# 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} % ')

# # Classification report
# report = classification_report(pred, y_test)
# print(report)

# # ROC Curve (For binary classification)
# 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/)
```