Pandas

In [ ]:
```
import pandas as po
```

In [ ]:
```
df = pd.DataFrame()

df = pd.read_csv(path, sep=seperator)
# 파일에 헤더가 없으나, 로드된 데이터에는 넣고자 할 경우
df = pd.read_csv(path, sep=seperator, header=None, names=headers)
```

In [ ]:
```
# Column2를 기준으로 데이터 유형 개수 분석
df.groupby(['Column1', 'Column2', ...])['Column2'].count()

# df에서 'Column' 제거
df.drop('Column', axis=int, inplace=True)

# df에서 각 행에 함수 적용
df['newColumn'] = df['Column'].apply(func)

# Location
df.iloc[index]
df.iloc[1, -1]
df.loc[condition]
df.loc[['Column1', 'Column2', ...]]
```

Scikit-Learn

In [ ]:
```
from sklearn.metrics import accuracy_score
```

Preprocessing

In [ ]:
```
from sklearn import preprocessing


le = preprocessing.LabelEncoder()

# 범주형 변수에 숫자 할당
le = le.fit(df['Categorical Column'])
df['Categorical Column'] = le.transform(df['Categorical Column'])
```

Preparing Training

In [ ]:
```
from sklearn.model_selection import train_test_split


X_train, X_test, y_train, y_test = train_test_split(
    X,
    y,
    test_size=0.2,
    random_state=11
)
```

Classification

In [ ]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression


dt_clf = DecisionTreeClassifier(random_state=1)
rf_clf = RandomForestClassifier(random_state=1)
lr_clf = LogisticRegression()
```

In [ ]:
```
# DecisionTreeClassifier 학습/예측/평가
dt_clf.fit(X_train, y_train)
dt_pred = dt_clf.predict(X_test)
print(
    'DecisionTreeClassifier 정확도: {0:.4f}'
    .format(accuracy_score(y_test, dt_pred))
)

# RandomForestClassifier 학습/예측/평가
rf_clf.fit(X_train, y_train)
rf_pred = rf_clf.predict(X_test)
print(
    'RandomForestClassifier 정확도:{0:.4f}'
    .format(accuracy_score(y_test, rf_pred))
)

# LogisticRegression 학습/예측/평가
lr_clf.fit(X_train , y_train)
lr_pred = lr_clf.predict(X_test)
print(
    'LogisticRegression 정확도: {0:.4f}'
    .format(accuracy_score(y_test, lr_pred))
)
```

In [ ]:
```
from sklearn.model_selection import KFold


clf = (Classifier)
kfold = KFold(n_splits=5)
for train_index, test_index in kfold.split(X):
    print(train_index, test_index)
    # [1667 1668 1669 ... 9997 9998 9999] [   0    1    2 ... 1664 1665 1
    # [   0    1    2 ... 9997 9998 9999] [1667 1668 1669 ... 3331 3332 3
    # [   0    1    2 ... 9997 9998 9999] [3334 3335 3336 ... 4998 4999 5
    # [   0    1    2 ... 9997 9998 9999] [5001 5002 5003 ... 6665 6666 6
```

```
    # [   0    1    2 ... 9997 9998 9999] [6668 6669 6670 ... 8331 8332 8
    # [   0    1    2 ... 8331 8332 8333] [8334 8335 8336 ... 9997 9998 9

    X_train, X_test = X.values[train_index], X.values[test_index]
    y_train, y_test = y.values[train_index], y.values[test_index]

    clf.fit(X_train, y_train)
    pred = clf.predict(X_test)
    # accur = accuracy_score(y_test, pred)
```

https://scikit-learn.org/stable/modules/cross_validation.html#cross-validation

In [ ]:
```
from sklearn.model_selection import cross_val_score


cross_val_score(clf, X, y)
```

In [ ]:
```
from sklearn.model_selection import GridSearchCV


# Candidates: These will be evaluated
params = {
    'n_estimators':[100],
    'max_depth' : [6, 8, 10, 12],
    'min_samples_leaf' : [8, 12, 18 ],
    'min_samples_split' : [8, 16, 20]
}
rf_clf = RandomForestClassifier(random_state=0, n_jobs=-1)
grid_cv = GridSearchCV(rf_clf, param_grid=params , cv=2, n_jobs=-1)
grid_cv.fit(X_train , y_train)

# 최적 파라미터 평가 결과 사용
best_params = grid_cv.best_params_
print(f'Best Hyperparams: {best_params}')
print(f'Best Accuracy: {grid_cv.best_score_}')
'''
Best Hyperparams: {'max_depth': 10, 'min_samples_leaf': 8, 'min_samples_s
Best Accuracy: 0.9172
'''

rf_clf = RandomForestClassifier(**best_params)
rf_clf.fit(X, y)
pred = rf_clf.predict(X_test)


# 특성값들의 중요도
ftr_importances_values = rf_clf.feature_importances_
ftr_importances = pd.Series(ftr_importances_values, index=X_train.columns
```

Metrics

In [ ]:
```
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.metrics import precision_score, recall_score
from sklearn.metrics import f1_score, roc_auc_score


## `y` and `pred` is known: y and h_hat
confusion = confusion_matrix(y_test, pred)
accuracy = accuracy_score(y_test, pred)
precision = precision_score(y_test, pred)
recall = recall_score(y_test, pred)
f1 = f1_score(y_test, pred)

# ROC-AUC: https://scikit-learn.org/stable/modules/generated/sklearn.metr
roc_auc = roc_auc_score(y_test, y_score)
```

Ensemble

In [ ]:
```
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score


X_train, X_test, y_train, y_test

knn_clf = KNeighborsClassifier(n_neighbors=4)
rf_clf = RandomForestClassifier(n_estimators=100, random_state=0)
dt_clf = DecisionTreeClassifier()
ada_clf = AdaBoostClassifier(n_estimators=100)

knn_clf.fit(X_train, y_train)
rf_clf.fit(X_train, y_train)
dt_clf.fit(X_train, y_train)
ada_clf.fit(X_train, y_train)

knn_pred = knn_clf.predict(X_test)
rf_pred = rf_clf.predict(X_test)
dt_pred = dt_clf.predict(X_test)
ada_pred = ada_clf.predict(X_test)

accuracy_score(y_test, knn_pred)
accuracy_score(y_test, rf_pred)
accuracy_score(y_test, dt_pred)
accuracy_score(y_test, ada_pred)
```

In [ ]:
```
from sklearn.linear_model import LogisticRegression


# Stacking Ensemble
pred = np.array(knn_pred, rf_pred, dt_pred, ada_pred)
pred = np.transpose(pred)
# (114, 4): 4개 모델의 pred값 병합

lr_clf = LogisticRegression()
lr_clf.fit(pred, y_test)
lr_clf_pred = lr_clf.predict(pred)
accuracy_score(y_test, lr_clf_pred)
```

Oversampling and Undersampling

In [ ]:
```
from imblearn.over_sampling import SMOTE


smote = SMOTE()
X_over, y_over =  smote.fit_resample(X, y)
```