summary note

Agenda

1. 데이터 전처리

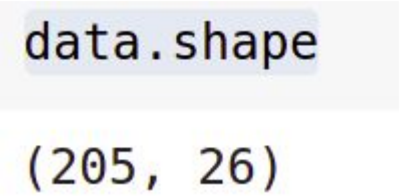
2. 모델 학습

3. 평가

데이터 전처리

1. 데이터 차원확인

```
data.shape     #(데이터수,차원수)
```



```
data.shape
```

```
(205, 26)
```

2. 데이터 수치변경 (관련예제 autoprice regression)

```
data.replace("?", np.nan, inplace = True)     # 데이터값 "?"을 np.nan으로 변경
```

데이터 전처리

3. Column 데이터 조작(manipulation) : apply 사용

```
# apply 함수를 사용. 장점 : 함수만 변경하면, 데이터값을 변환하는 다양한 상황에서
호환성있게 사용할 수 있다.

def majority(x):
    if x > 17:
        return True
    else:
        return False

cols = ['normalized_losses', 'bore', 'stroke', 'horse_power', 'peak_rpm', 'price']

for col in cols:
  data[col] = data[col].apply(majority)
```

데이터 전처리

4. drop, dropna

```
# 결측치 있는 데이터(row) 제거. 관련예제 autoprice regression
x = data['price'].dropna(axis=0)
data = data.loc[x.index]
data.reset_index(drop=True, inplace=True)
```

```
# column name으로 column 제거. 예제 titanic classification
x.drop(['name','ticket', 'boat', 'body', 'home.dest'], axis=1, inplace=True)
```

데이터 전처리

5. 학습, 테스트 데이터 스플릿

사용예시 :

x_train, x_test, y_train, y_test = train_test_split(data, test_size, random_state, shuffle, stratify)

```
# train,validation, test 분리. 관련예제 titanic classification

x = data.drop(['label'], axis=1)
y = data['label']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=2021)
x_train, x_valid, y_train, y_valid = train_test_split(x_train, y_train, test_size=2/8,
random_state = 2021)
```

데이터 전처리

6. 결측값 imputing

```
# 결측값 imputing. 관련예제 autoprice regression

from sklearn.impute import SimpleImputer

imputer_num = SimpleImputer(strategy='mean') # 평균 imputing 객체생성
imputer_ca = SimpleImputer(strategy='most_frequent') # 최빈값 imputing 객체생성

num_cols = ['normalized_losses', 'bore', 'stroke', 'horse_power', 'peak_rpm'] # NaN이 있었던 numeric columns
ca_cols = ['num_of_doors'] # NaN이 있었던 categorical columns

x_train[ca_cols] = imputer_ca.fit_transform(x_train[ca_cols]) #학습셋에 fit + transform
x_train[num_cols] = imputer_num.fit_transform(x_train[num_cols]) #학습셋에 fit + transform

x_test[ca_cols] = imputer_ca.transform(x_test[ca_cols]) #학습셋의 파라미터로 transform
x_test[num_cols] = imputer_num.transform(x_test[num_cols]) ##학습셋의 파라미터로 transform
```

데이터 전처리

7. categorical feature 더미화
   x_dummies= pd.get_dummies(x, columns=categorical_columns, drop_first)

```
# categorical feature 더미화. 관련예제 titanic classification

x_train = pd.get_dummies(x_train, columns=['pclass', 'cabin', 'embarked', 'fs', 'sex'],
drop_first= True)
x_valid = pd.get_dummies(x_valid, columns=['pclass', 'cabin', 'embarked', 'fs', 'sex'],
drop_first= True)
x_test = pd.get_dummies(x_test, columns=['pclass', 'cabin', 'embarked', 'fs', 'sex'],
drop_first= True)

x_valid = x_valid[x_train.columns] #학습셋과 동일한 column이 되도록 변경
x_test = x_test[x_train.columns] #학습셋과 동일한 column이 되도록 변경
```

Agenda

1. 데이터 전처리

2. 모델 학습

3. 평가

모델 학습

1. Linear regression

```
#관련예제 autoprice regression

from sklearn.linear_model import LinearRegression

# Linear regression 학습
lr = LinearRegression()
lr.fit(x_train, y_train)

# 특정 column의 데이터값을 기준으로 Linear regression 학습
lr_turbo = LinearRegression(); lr_std = LinearRegression();
x_train_t = x_train[x_train['aspiration_turbo']==1]; y_train_t = y_train[x_train['aspiration_turbo']==1]
x_train_s = x_train[x_train['aspiration_turbo']==0]; y_train_s = y_train[x_train['aspiration_turbo']==0]

lr_std.fit(x_train_s, y_train_s)
lr_turbo.fit(x_train_t, y_train_t)
```

모델 학습

2. Decision tree : scikit-learn

```
#관련예제 titanic classification

from sklearn.tree import DecisionTreeClassifier

trees = []

for i in range(1, 31):
    dtr = DecisionTreeClassifier(max_depth=3, min_samples_leaf=i)
    dtr.fit(x_train, y_train)
    trees.append(dtr)
```

모델 학습

3. RandomForest : scikit-learn

```
#관련예제 random forest

from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=50,
                             min_samples_split=10, max_depth=None,criterion='gini' )
rfc.fit(X_train, y_train)
```

모델 학습

3. PCA & K-means

```
#관련예제 PCA + K-means

from sklearn.decomposition import PCA
pca = PCA(n_components=None)
dfx_pca = pca.fit(dfx)

dfx_trans = pca.transform(dfx) #새로운 PC들로 데이터 변환

from sklearn.cluster import KMeans
kms = KMeans(3, random_state = 2021).fit(dfx_trans.loc[:,:4]) #변환된 데이터에서 PC 1~4로 클러스터링
cluster = kms.predict(dfx_trans.loc[:,:4])
print(cluster)
```

Agenda

1. 데이터 전처리

2. 모델 학습

3. 평가

평가(metric)

1. MAE, MSE (regression)

```
#관련예제 autoprice regression

from sklearn.metrics import mean_squared_error as MSE
from sklearn.metrics import mean_absolute_error as MAE

y_pred = your_model.predict(x_test)
mae = MAE(y_test, y_pred) #MAE
mse = MSE(y_test, y_pred) #MSE
rmse = MSE(y_test, y_pred)**.5 #RMSE
```

평가(metric)

2. accuracy, recall, precision, sensitivity, specificity (classification)

#관련예제 titanic classification

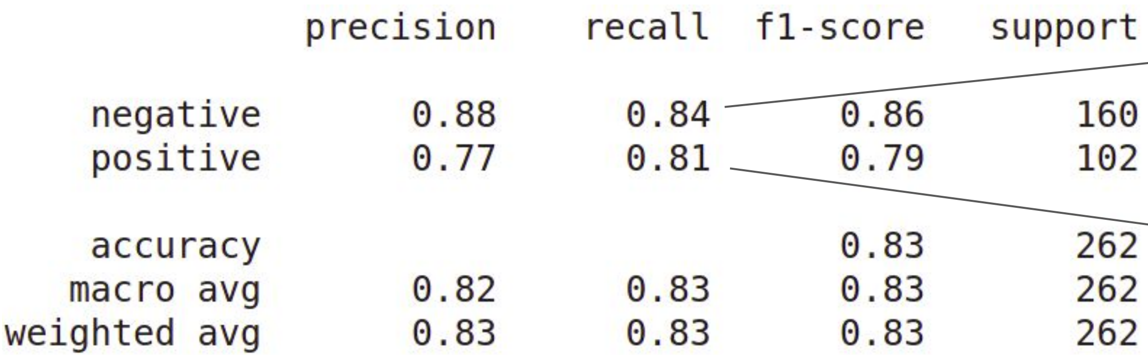
```
from sklearn.metrics import accuracy_score as ACC

y_pred = your_model.predict(x_test)
acc = ACC(y_test, y_pred)
```

평가(metric)

#관련예제 titanic classification

```
from sklearn.metrics import classification_report
best_tree = trees[24]
y_pred = best_tree.predict(x_test)
print(classification_report(y_test, y_pred, target_names=['negative', 'positive']))
```

Note that in binary classification,
recall of the positive class is also known as “sensitivity”;
recall of the negative class is “specificity”.



```
              precision    recall  f1-score   support

    negative       0.88      0.84      0.86       160
    positive       0.77      0.81      0.79       102

    accuracy                           0.83       262
   macro avg       0.82      0.83      0.83       262
weighted avg       0.83      0.83      0.83       262
```

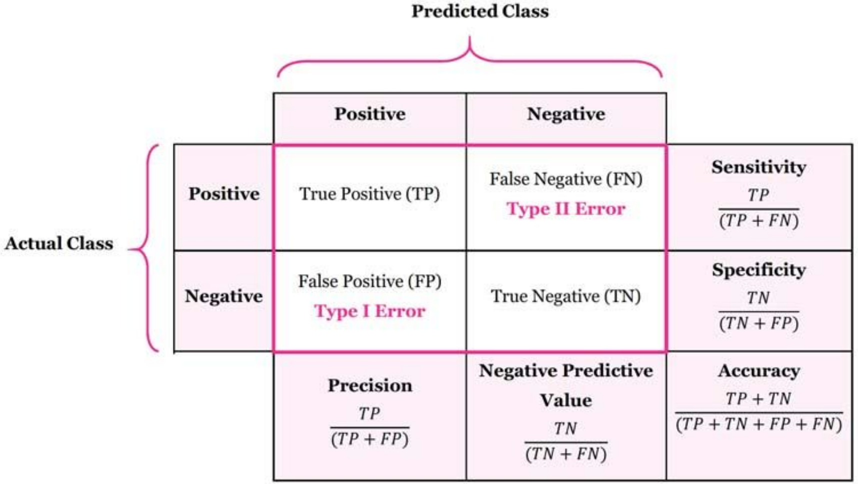
specificity

sensitivity

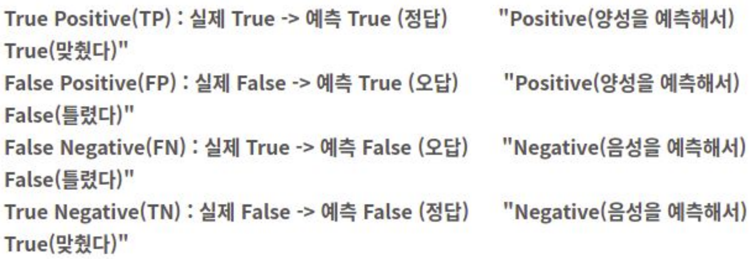
평가(metric)

#관련예제 random forest & titanic classification

```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(label, prediction)
```



		Predicted Class		
		Positive	**Negative**	
Actual Class	**Positive**	True Positive (TP)	False Negative (FN) **Type II Error**	**Sensitivity** $\frac{TP}{(TP+FN)}$
	Negative	False Positive (FP) **Type I Error**	True Negative (TN)	**Specificity** $\frac{TN}{(TN+FP)}$
		Precision $\frac{TP}{(TP+FP)}$	**Negative Predictive Value** $\frac{TN}{(TN+FN)}$	**Accuracy** $\frac{TP+TN}{(TP+TN+FP+FN)}$



True Positive(TP) : 실제 True -> 예측 True (정답) "Positive(양성을 예측해서) True(맞췄다)"

False Positive(FP) : 실제 False -> 예측 True (오답) "Positive(양성을 예측해서) False(틀렸다)"

False Negative(FN) : 실제 True -> 예측 False (오답) "Negative(음성을 예측해서) False(틀렸다)"

True Negative(TN) : 실제 False -> 예측 False (정답) "Negative(음성을 예측해서) True(맞췄다)"

Thank you!