22-2 기계학습기초

경진대회 최종 발표

인공지능학부 215001 서가연

01. 머신러닝 경진대회

AutoML

	Description	Value
0	Session id	406
1	Target	move_out
2	Target type	Binary
3	Original data shape	(60832, 18)
4	Transformed data shape	(104777, 18)
5	Transformed train set shape	(92610, 18)
6	Transformed test set shape	(12167, 18)
7	Numeric features	17
8	Preprocess	True
9	Imputation type	simple
10	Numeric imputation	mean
11	Categorical imputation	constant
12	Low variance threshold	0
13	Fix imbalance	True
14	Fix imbalance method	SMOTE
15	Fold Generator	StratifiedKFold
16	Fold Number	10
17	CPU Jobs	-1

drop

best_5 = compare_models(sort='AUC', n_select=5)					
	Model	Accuracy	AUC		
lr	Logistic Regression	0.9802	0.9982		
lightgbm	Light Gradient Boosting Machine	0.9847	0.9962		
qda	Quadratic Discriminant Analysis	0.9507	0.9890		
ada	Ada Boost Classifier	0.9336	0.9857		
gbc	Gradient Boosting Classifier	0.9320	0.9843		
rf	Random Forest Classifier	0.9642	0.9821		
lda	Linear Discriminant Analysis	0.8280	0.9810		
et	Extra Trees Classifier	0.9504	0.9706		
nb	Naive Bayes	0.7507	0.9135		
dt	Decision Tree Classifier	0.9766	0.8805		
knn	K Neighbors Classifier	0.9024	0.8754		
dummy	Dummy Classifier	0.0485	0.5000		
svm	SVM - Linear Kernel	0.9789	0.0000		
ridge	Ridge Classifier	0.8280	0.0000		

FILLNA

best_5 = compare_models(sort='AUC', n_select=5)				
	Model	Accuracy	AUC	
lr	Logistic Regression	1.0000	1.0000	
dt	Decision Tree Classifier	1.0000		
qda	Quadratic Discriminant Analysis	0.9999	1.0000	
ada	Ada Boost Classifier	0.9757	1.0000	
gbc	Gradient Boosting Classifier	0.9967	1.0000	
lightgbm	Light Gradient Boosting Machine	1.0000	1.0000	
rf	Random Forest Classifier	0.9952	0.9996	
et	Extra Trees Classifier	0.9888	0.9977	
lda	Linear Discriminant Analysis	0.8275	0.9828	
nb	Naive Bayes	0.7537	0.9222	
knn	K Neighbors Classifier	0.9119	0.9005	
dummy	Dummy Classifier	0.0485	0.5000	
svm	SVM - Linear Kernel	1.0000	0.0000	
ridge	Ridge Classifier	0.8275	0.0000	

01. 머신러닝 경진대회

SMOTE 오버샘플링

```
### SMOTE 셀플링 추가
from sklearn.datasets import make_classification
from sklearn.decomposition import PCA
from imblearn.over_sampling import SMOTE

# 모델설정
smote = SMOTE(random_state=42)
X_train_over, y_train_over = smote.fit_sample(X_train, y_train)
print("SMOTE 적용 전 학습용 피처/레이블 데이터 세트 : ", X_train.shape, y_train.shape)
print("SMOTE 적용 후 학습용 피처/레이블 데이터 세트 :', X_train_over.shape, y_train_over.shape)
print("SMOTE 적용 후 값의 분포 :\n', pd.Series(y_train_over).value_counts())

SMOTE 적용 전 학습용 피처/레이블 데이터 세트 : (60832, 18) (60832,)
SMOTE 적용 후 학습용 피처/레이블 데이터 세트 : (115764, 18) (115764,)
SMOTE 적용 후 값의 분포 :
1 57882
0 57882
Name: move_out, dtype: int64
```

XGBOOST

Complete - 24d ago

가중치 동결

```
def build_top(base):
    x = base.output
    x = GlobalAveragePooling2D()(x)
   x = Dense(512, activation='relu')(x)
   x = Dropout(0.2)(x)
   x = Dense(256, activation='relu')(x)
    x = Dropout(0.2)(x)
   x = Dense(128, activation='relu')(x)
   outputs = Dense(len(labels), activation='softmax')(x)
   return Model(inputs=base.input, outputs=outputs)
def setup model(model. base):
    for layer in base.layers:
        layer.trainable = False
    modeL.compile(
        loss='categorical crossentropy',
        optimizer=Adam(0.0003),
        metrics=['accuracy']
```

loss: 0.4414 - accuracy: 0.8594 val loss: 0.4803 - val accuracy: 0.8482

trainable param 수 증가

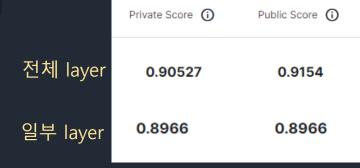
Total params: 23,987,860 Trainable params: 3,126,380

Non-trainable params: 20,861,480



Total params: 23,987,860

Trainable params: 23,933,332 Non-trainable params: 54,528



가중치 동결 해제

```
# 가중치 동결 해제 => fine_tuning

def tuned model(model):
    for layer in model:
        layer.trainable = True

model.compile(
        loss='categorical_crossentropy',
        # 학습들을 더 작게 설정함
        optimizer=Adam(0.00001),
        metrics=['accuracy']
)

tuned_model(model)
```

Xception



loss: 0.2443 - accuracy: 0.9208

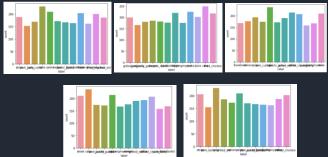
val_loss: 0.3751 - val_accuracy: 0.8879

모델 : EfficientNetV2M

```
base_model = EfficientNetV2M(include_top=False, input_tensor=Input(shape=(224, 224, 3)), weights='imagenet', classes=12)
base_model.trainable = False # 가중치 동결 후 학습

model = tf.keras.models.Sequential([
    base_model,
    GlobalAveragePooling2D(),
    BatchNormalization(),
    Dense(512, activation='relu'),
    Dropout(0.2),
    Dense(256, activation='relu'),
    Dropout(0.2),
    Dense(128, activation='relu'),
    Dense(129, activation='relu'),
    Dense(120, activation='relu'),
    Dense(121, activation='softmax'), # output

])
```



freeze & fold

```
# 5fold
for iter in range(5):
   print(f"\n=======\n")
   train generator = train datagen.flow from dataframe(train list[iter],
                                                directory = '/train',
                                                x_col='image',
                                                y col='label',
                                                batch_size=32,
                                                color mode= 'rgb',
                                                target_size=(height, width), shuffle=True)
   valid generator = valid datagen.flow from dataframe(valid list[iter],
                                                directory = '/train',
                                                x_col='image',
                                                y col='label',
                                                batch_size=32,
                                                color mode= 'rgb',
                                                target_size=(height, width), shuffle=True)
   model.compile(optimizer=Adam(0.001), loss='categorical crossentropy', metrics=['accuracy'])
   print("\n====== train start! ======\n")
   history = model.fit(train_generator,
                       epochs=10.
                       callbacks=callbacks,
                       validation data=valid generator)
```

unfreeze & fold

```
for iter in range(5):
   print(f"\n=======(iter+1)번째 학습======\n")
   for layer in model.layers[:-iter*2 + 20]:
       layer.trainable = True
   train generator = train datagen.flow from dataframe(train list[iter],
                                                 directory = '/train',
                                                 x_col='image',
                                                 y col='label',
                                                 batch_size=32,
                                                 color mode= 'rgb',
                                                 target_size=(height, width), shuffle=True)
   valid generator = valid datagen.flow from dataframe(valid list[iter],
                                                 directory = '/train',
                                                 x col='image',
                                                 y_col='label',
                                                 batch_size=32,
                                                 color_mode= 'rgb',
                                                 target_size=(height, width), shuffle=True)
   model.compile(optimizer=Adam(0.00035), loss='categorical_crossentropy', metrics=['accuracy'])
   print("\n====== train start! ======\n")
   history = model.fit(train_generator,
                       epochs=20,
                       callbacks=callbacks,
                       validation data=valid generator)
```

Ø	eff300_tuned_fold (1) Complete · 7d ago	.csv Target_size=(300, 300)	0.93058	0.93926
0	eff_tuned_fold.csv Complete · 7d ago	Unfreeze & fold	0.9219	0.93853
0	eff_fold.csv Complete · 7d ago	freeze & fold	0.90238	0.89949

감사합니다