02. Deep Learning

권형근 이세나 이지현 조상현

CONTENTS

PART1

데이터 전처리

1. 데이터 전처리

1. 데이터 전처리

● 결측치 처리 및 데이터 분리

결측치 처리

```
data = data.fillna(-1)
```

```
data.drop("Unnamed: 0",axis=1,inplace=True)
```

데이터 분리 및 Index 초기화

```
from sklearn.model_selection import train_test_split
```

```
trn, val = train_test_split(data, test_size=0.2,shuffle=True, stratify=data["class"], random_state=42)
```

```
valid = data.iloc[val.index,:]
train = data.iloc[trn.index, :]
```

```
train = train.reset_index(drop=True)
valid = valid.reset_index(drop=True)
```

1. 데이터 전처리

- **결측치 처리 및 데이터 분리**

결측치 처리

```
data = data.fillna(-1)
```

```
data.drop("Unnamed: 0",axis=1,inplace=True)
```

데이터 분리 및 Index 초기화

```
from sklearn.model_selection import train_test_split
```

```
trn, val = train_test_split(data, test_size=0.2,shuffle=True, stratify=data["class"], random_state=42)
```

```
valid = data.iloc[val.index,:]
train = data.iloc[trn.index, :]
```

```
train = train.reset_index(drop=True)
valid = valid.reset_index(drop=True)
```

1. 다른 값들과 다른 CLASS로 두기 위해(Tree 계열 모델)

2. 변수들의 값들의 분포가 넓기 때문에

1. 데이터 전처리

● 결측치 처리 및 데이터 분리

결측치 처리

```
data = data.fillna(-1)
```

```
data.drop("Unnamed: 0",axis=1,inplace=True)
```

데이터 분리 및 Index 초기화

```
from sklearn.model_selection import train_test_split
```

```
trn, val = train_test_split(data, test_size=0.2,shuffle=True, stratify=data["class"], random_state=42)
```

```
valid = data.iloc[val.index,:]
train = data.iloc[trn.index, :]
```

```
train = train.reset_index(drop=True)
valid = valid.reset_index(drop=True)
```

1. 데이터 전처리

- 레이블 인코딩

Target 변수 encoding

```
x_train = train.iloc[:,1:]
y_train = train["class"]
x_valid = valid.iloc[:,1:]
y_valid = valid["class"]
y_train = y_train.replace({"neg" : 0, "pos" : 1})
y_valid = y_valid.replace({"neg" : 0, "pos" : 1})
```

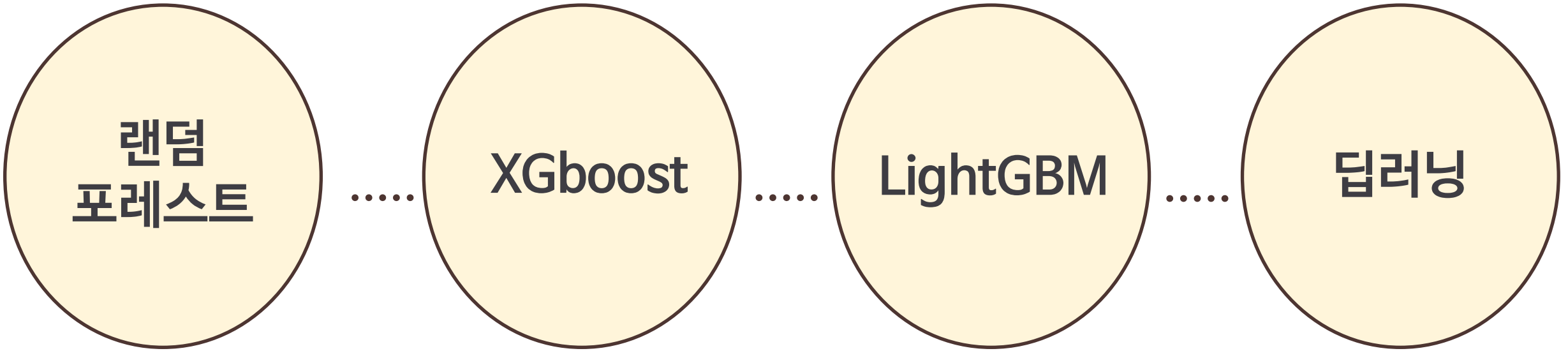
```
y_train
```

```
0         0
1         0
2         0
3         0
4         0
         ..
45595     0
45596     0
45597     0
45598     0
45599     0
Name: class, Length: 45600, dtype: int64
```

PART2

모델링

2. 모델링



2. 모델링

랜덤 포레스트

```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_jobs= 2, random_state=42)
rf.fit(x_train,y_train)
rf_pred = rf.predict(x_valid)
rf_pred
y_valid.values
```

```
cost(rf_pred,y_valid.values)
```

```
27720
```

XGboost

```
from xgboost import XGBClassifier
xgb = XGBClassifier(max_depth = 10, learning_rate=0.05,n_estimators=1600,
xgb.fit(x_train,y_train)
xgb_pred = xgb.predict(x_valid)
```

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=0,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.05, max_delta_step=0, max_depth=10,
              min_child_weight=1, missing=nan,

n_estimators=1600, n_jobs=0, num_parallel_tree=1, random_state=0,
reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
tree_method='gpu_hist', validate_parameters=1, verbosity=None)
```

```
cost(xgb_pred,y_valid.values)
```

```
19190
```

2. 모델링

LightGBM

```
from lightgbm import LGBMClassifier
lgb = LGBMClassifier(boosting_type='gbdt', num_leaves=1024, learning_rate=0.02,
                     n_estimators=1500, device = "gpu")
lgb.fit(x_train,y_train)
lgb_pred = lgb.predict(x_valid)
```

```
LGBMClassifier(device='gpu', learning_rate=0.02, n_estimators=1500,
               num_leaves=1024)
```

```
cost(lgb_pred,y_valid.values)
```

```
22200
```

2. 모델링

딥러닝

- 정규화

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train.iloc[:,:] = sc.fit_transform(x_train.iloc[:,:])
x_valid.iloc[:,:] = sc.transform(x_valid.iloc[:,:])
```

- 딥러닝 모델

```
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense, Dropout
def model_create() :
    model = Sequential()
    model.add(Dense(512, input_dim=x_train.shape[1], activation = "relu"))
    model.add(Dropout(0.5))
    model.add(Dense(384, activation = "relu"))
    model.add(Dropout(0.5))
    model.add(Dense(2, activation = "softmax"))
    model.compile(loss="sparse_categorical_crossentropy",
                  optimizer="adam", metrics=["sparse_categorical_accuracy"])
    return model
```

2. 모델링

딥러닝

- k-Fold

```
from sklearn.model_selection import KFold
from keras.callbacks import EarlyStopping, ReduceLROnPlateau, ModelCheckpoint
kf = KFold(n_splits=5, shuffle=True, random_state=41)
i = 1
# callbacks = [EarlyStopping(monitor='val_loss',patience=3,)]
callbacks = [EarlyStopping(monitor='val_loss',patience=5,),
             ModelCheckpoint(filepath='best_weight.hdf5', verbose=1,
                             save_best_only=True, mode='auto')]
```

- 모델 실행

```
preds_test = np.zeros((len(x_valid),2) ,dtype = np.float)
filepath = './best_weight.hdf5'
for train_index, valid_index in kf.split(x_train):
    X_train, X_valid = x_train.iloc[train_index, :], x_train.iloc[valid_index, :]
    Y_train, Y_valid = y_train[train_index] ,y_train[valid_index]
    model = model_create()
    model.fit(X_train, Y_train, batch_size=512, epochs=1000, verbose=1,
              validation_data=(X_valid, Y_valid), callbacks = callbacks)
    model.load_weights(filepath)
    preds_test_fold = model.predict(x_valid,batch_size = 256, verbose=1)
#     preds_test_fold = np.reshape(preds_test_fo
    preds_test += preds_test_fold
preds_test /= 5
```

```
cost(np.argmax(preds_test,axis=1),y_valid)
```

```
32310
```

2. 모델링

앙상블

```
final_pred = preds_test*0.25 + lgb_pred_proba*0.15 + xgb_pred_proba*0.6
```

```
cost(np.argmax(final_pred,axis=1),y_valid)
```

```
19190
```

PART3

결론

3. 결론

결과값

Accuracy Score

0.9748

Cost

155220

Confusion Matrix

18519	172
307	2

3. 결론

고찰

```
: final_pred =  xgb_pred_proba2*0.1 + lgb_pred_proba2*0.3 + preds_test2*0.6
```

Thank you